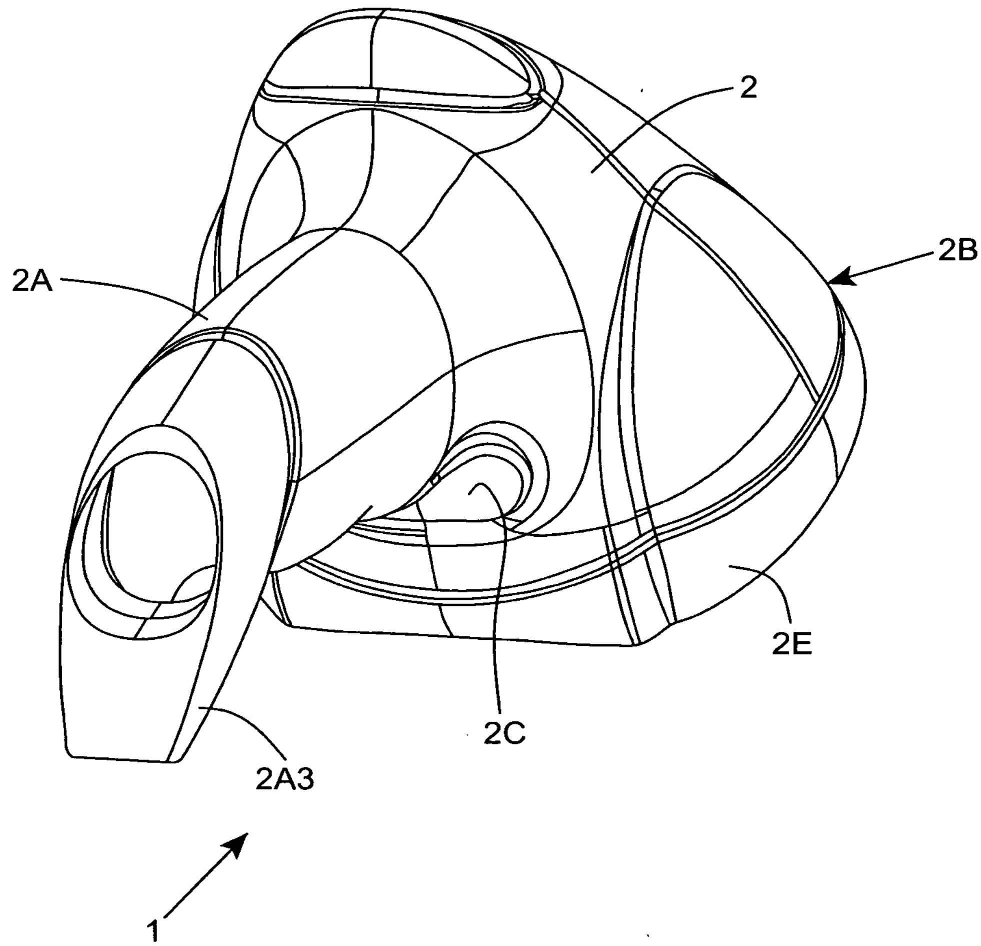


FIG. 1A

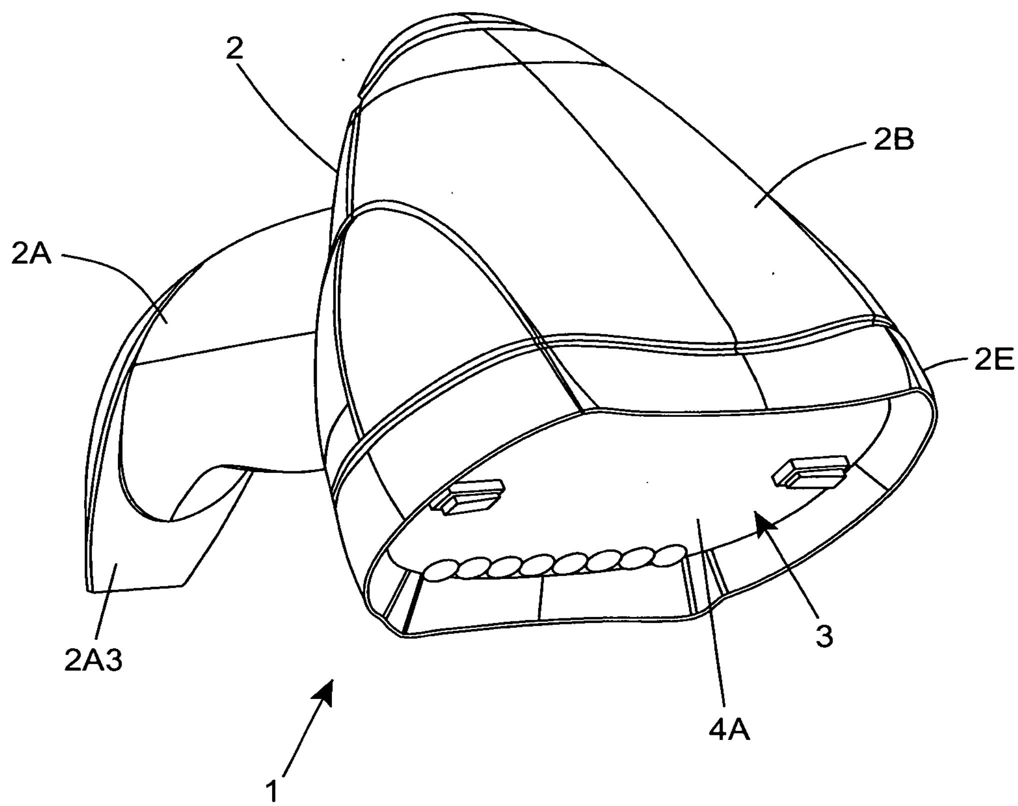


FIG. 1B

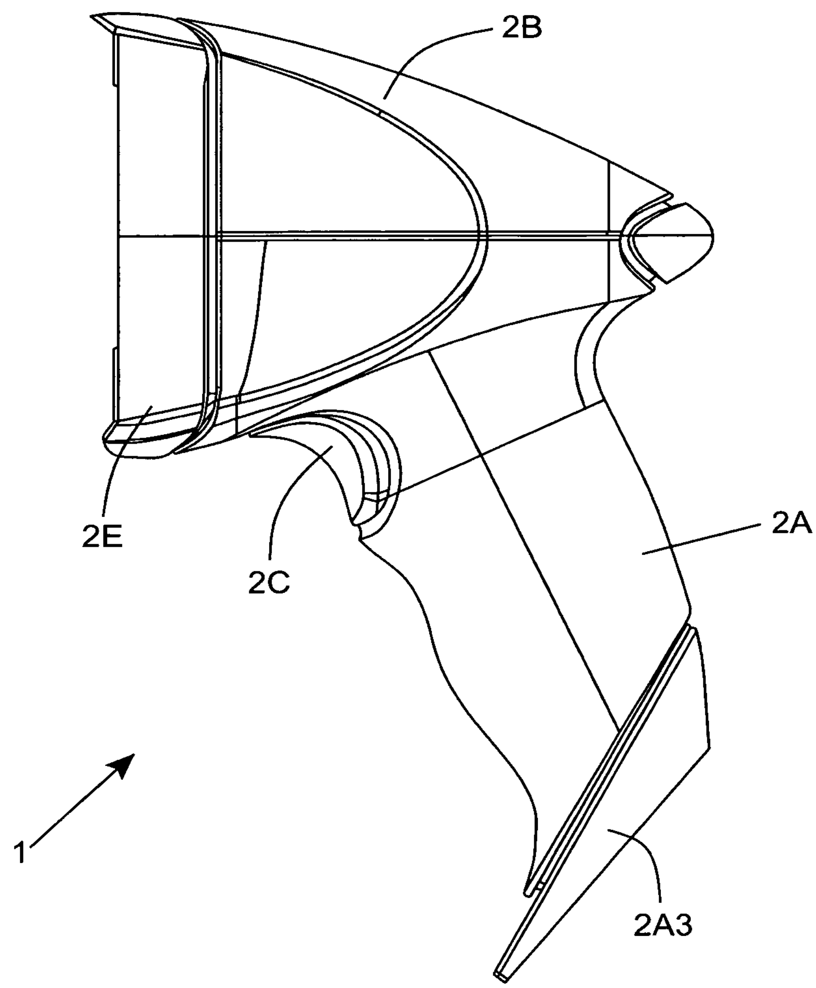


FIG. 1C

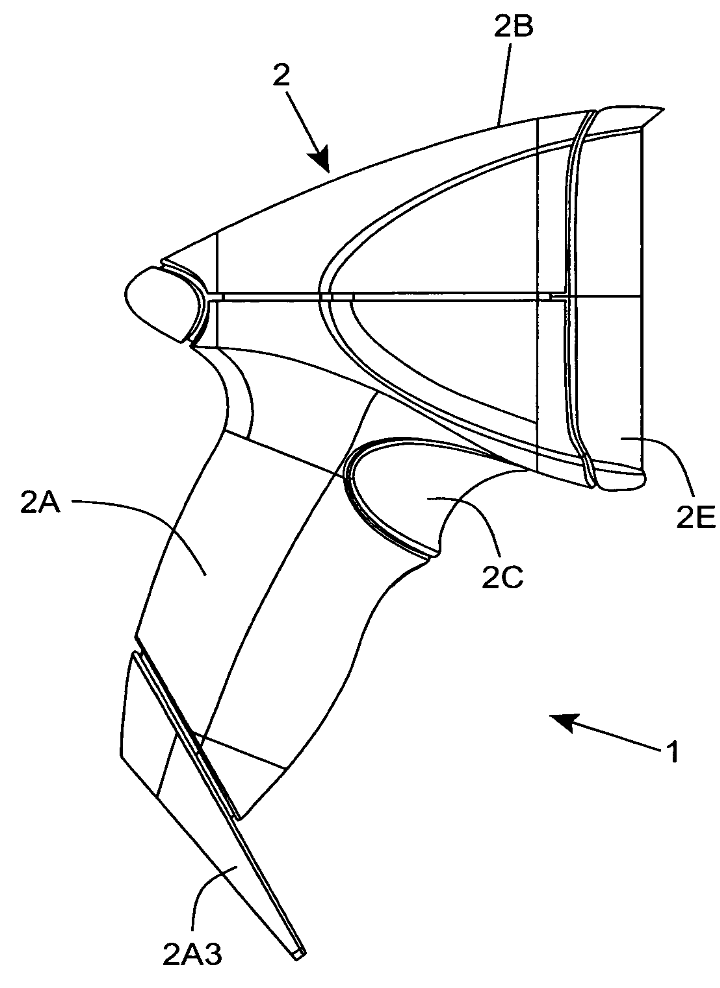


FIG. 1D

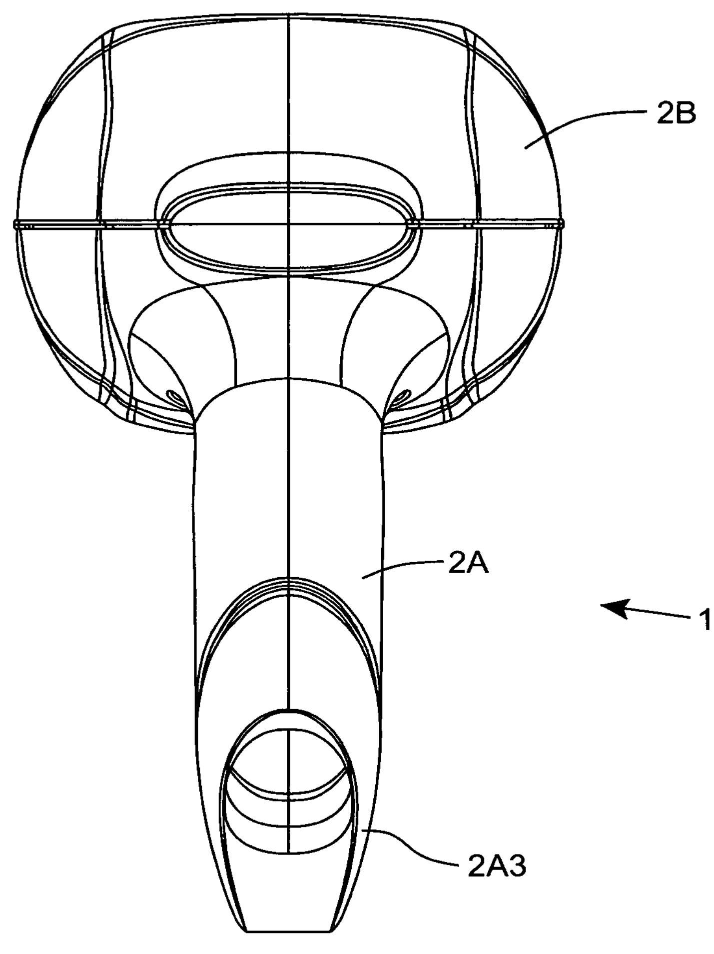


FIG. 1E

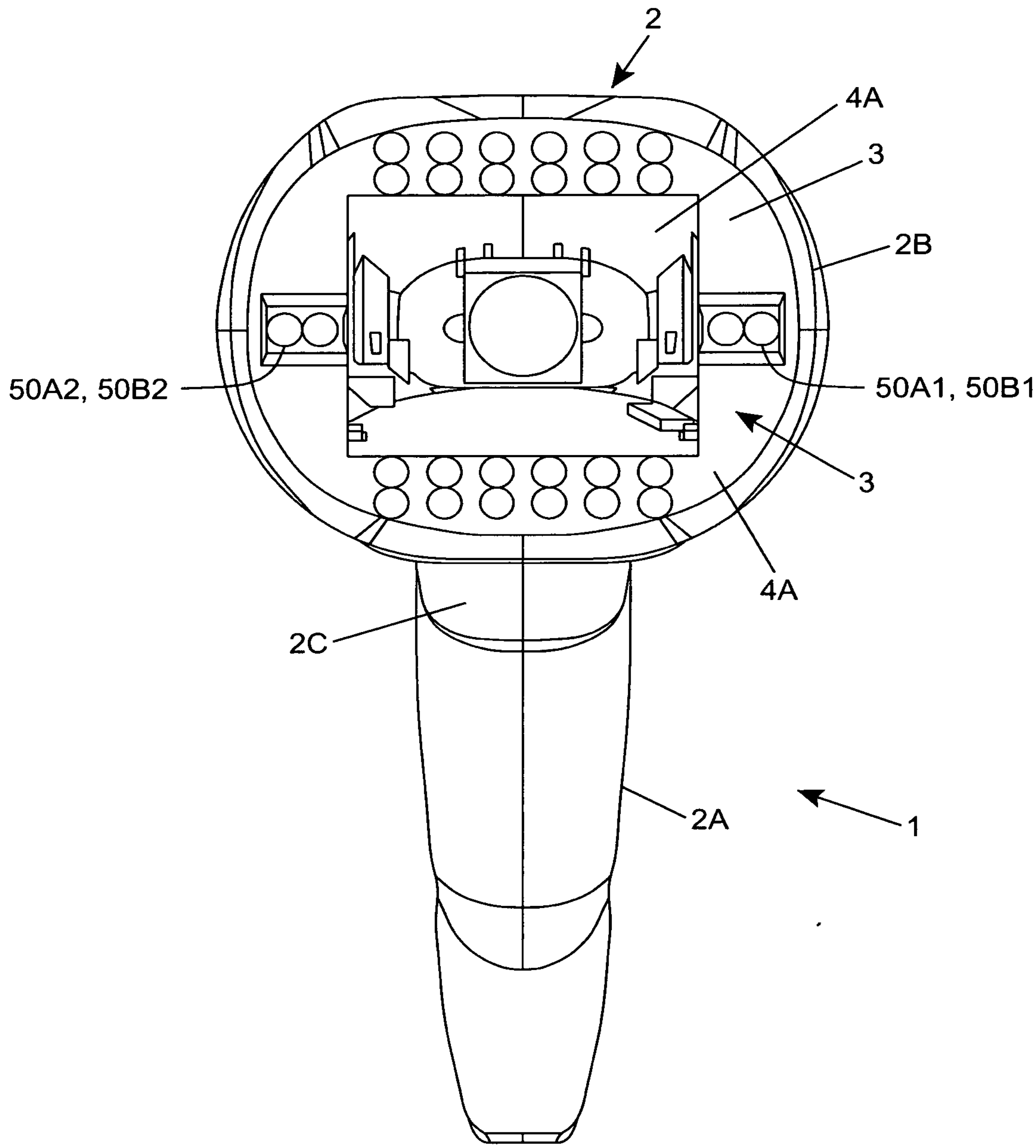


FIG. 1F

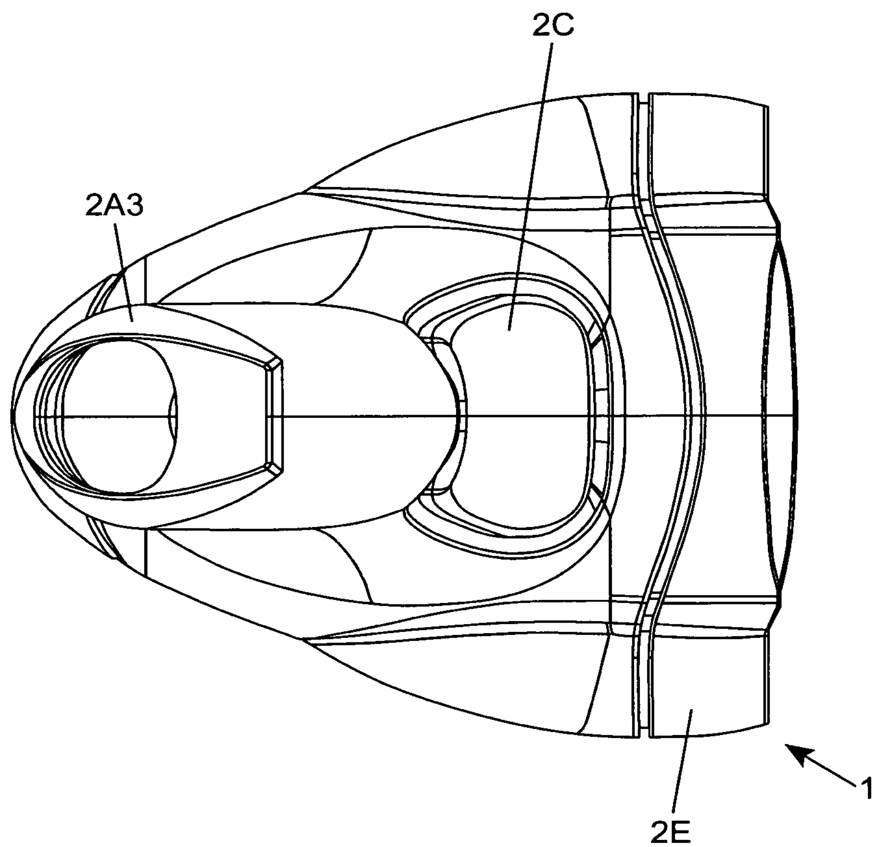


FIG. 1G

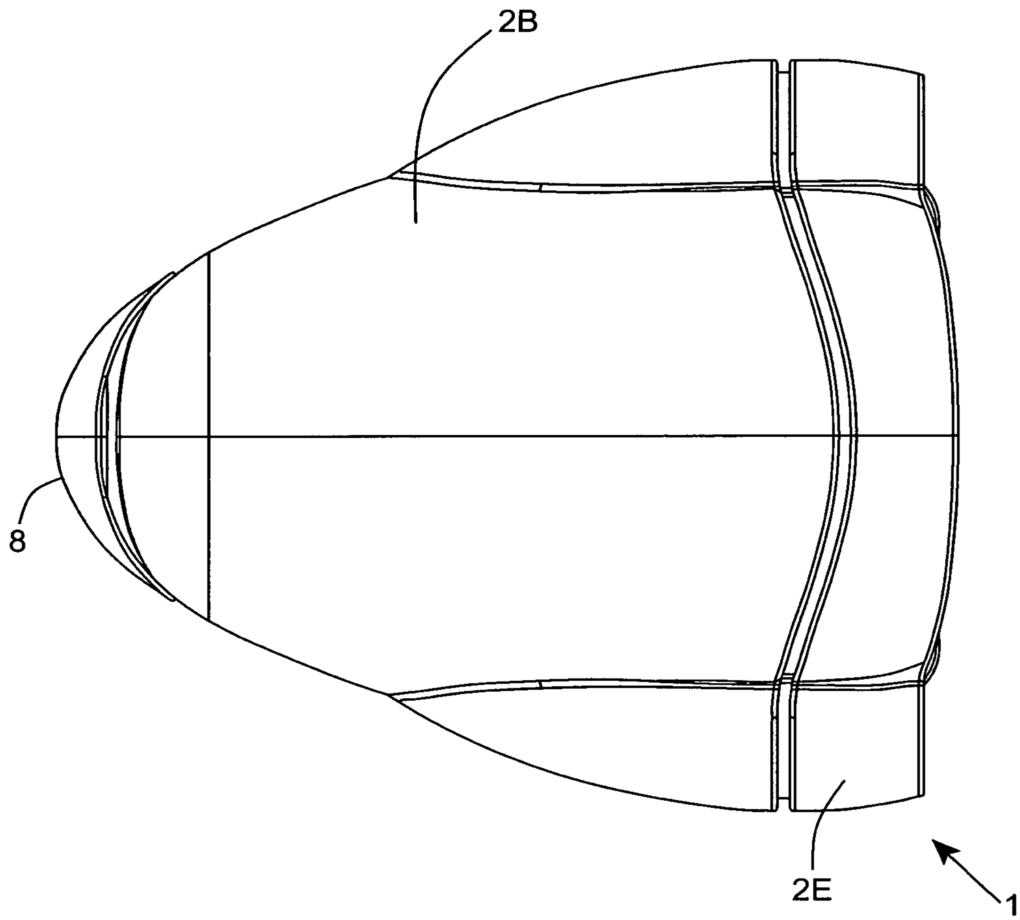


FIG. 1H

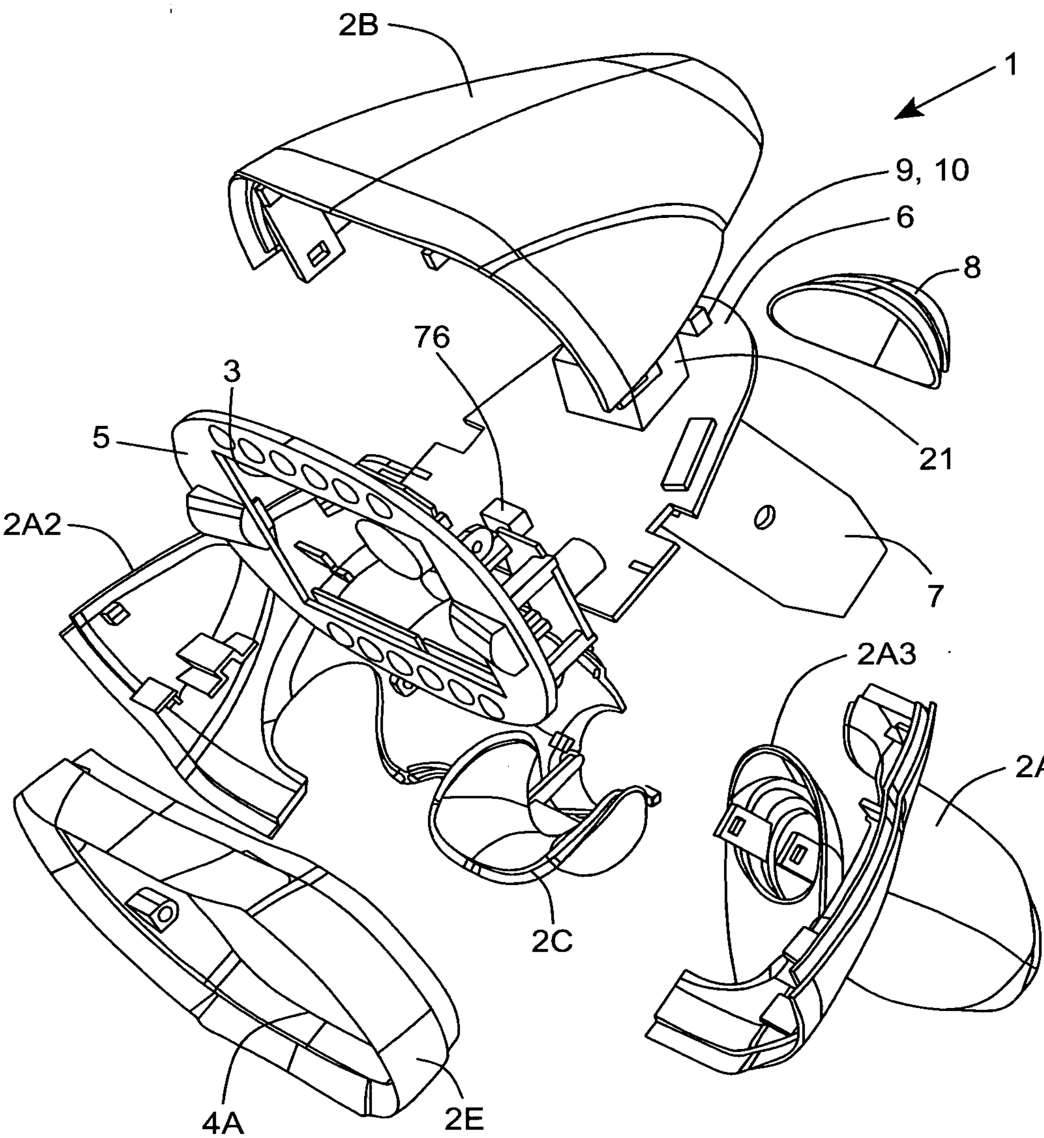


FIG. 1I

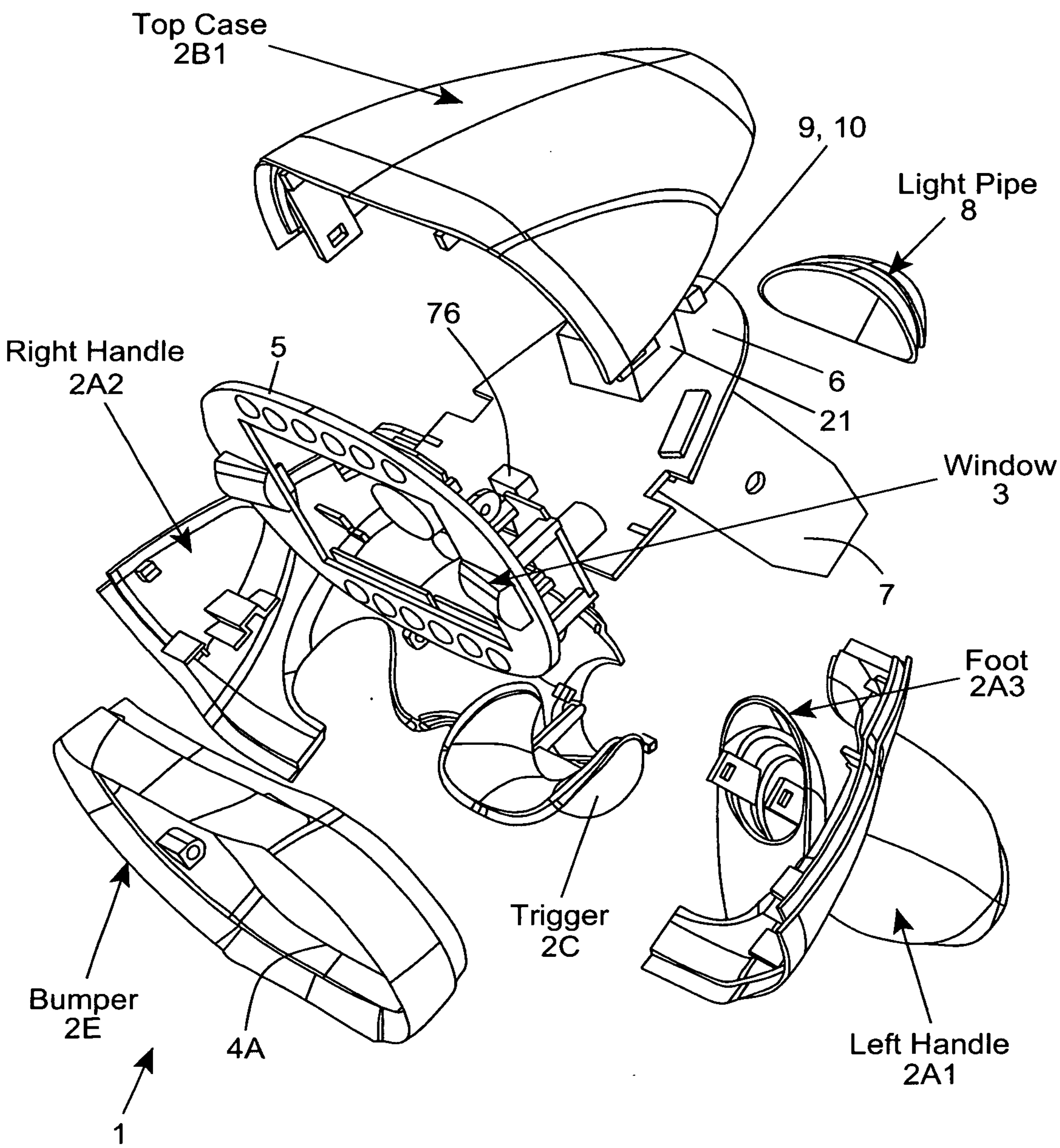

Top Case
2B1
9, 10
Light Pipe
8
76
Right Handle
2A2
5
6
21
Window
3
7
Foot
2A3
Trigger
2C
Bumper
2E
4A
1
Left Handle
2A1

FIG. 1J

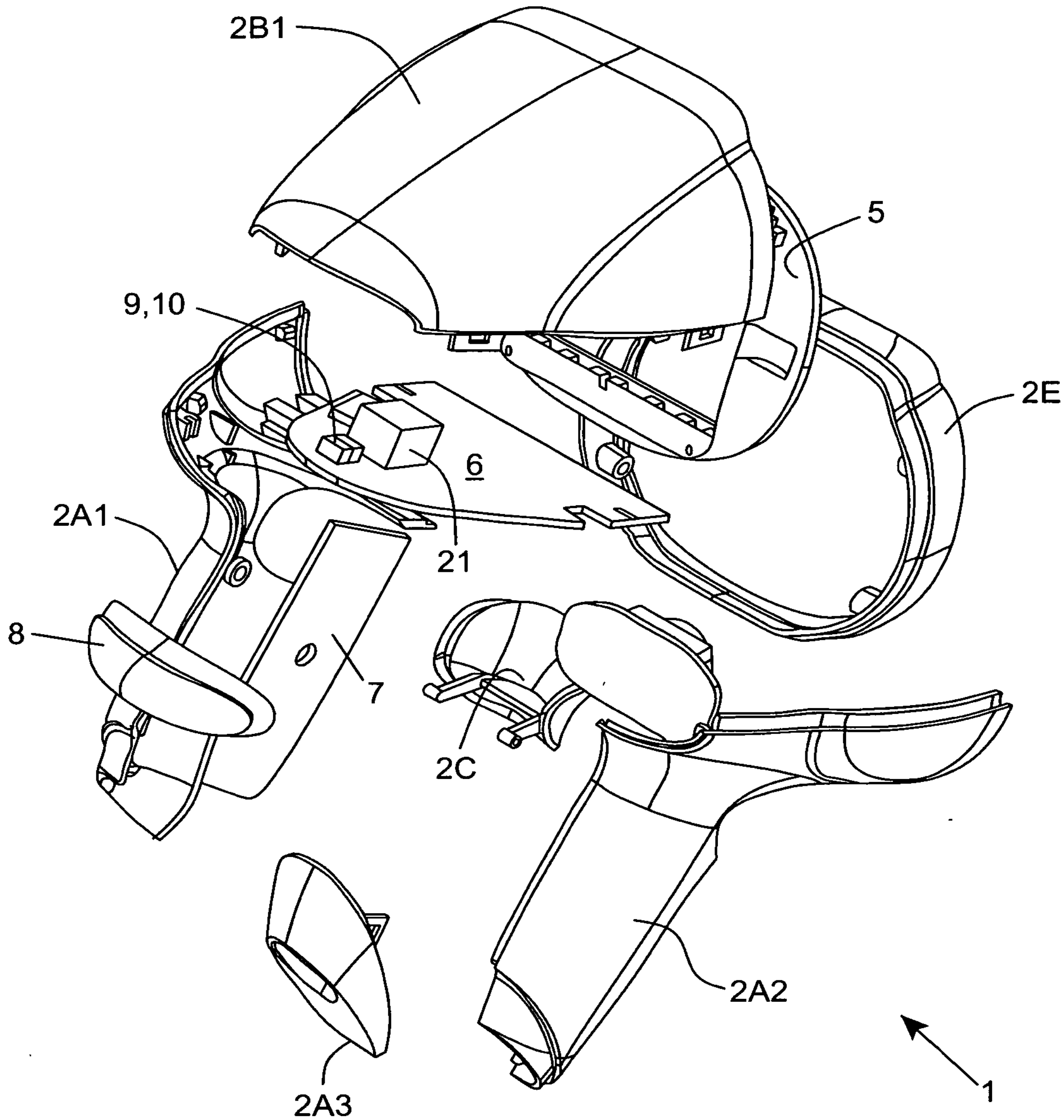
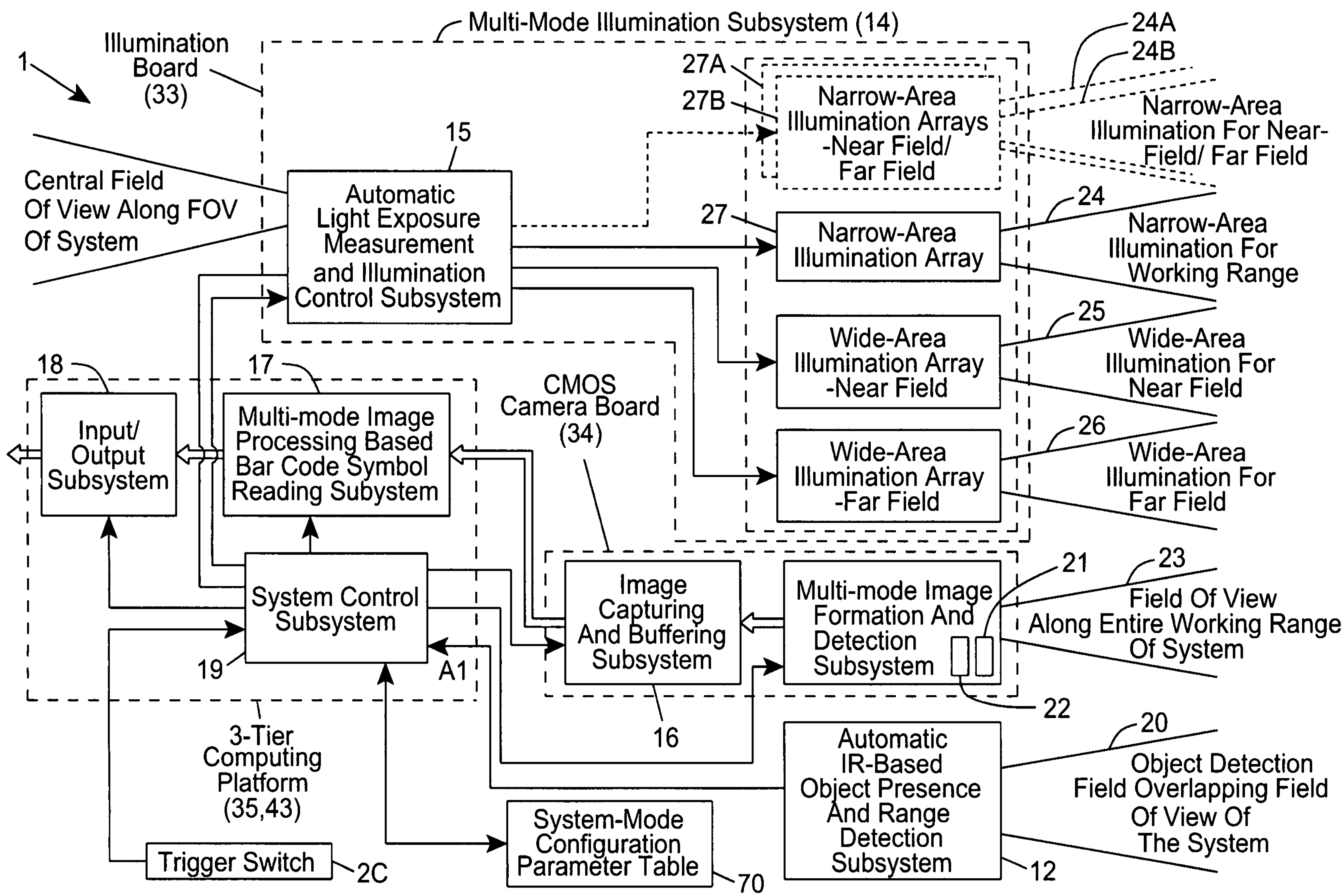


FIG. 1K



Multi-Mode Illumination Subsystem (14)
Illumination Board (33)
1
Central Field Of View Along FOV Of System
15
Automatic Light Exposure Measurement and Illumination Control Subsystem
27A
27B
Narrow-Area Illumination Arrays -Near Field/ Far Field
24A
24B
Narrow-Area Illumination For Near-Field/ Far Field
27
Narrow-Area Illumination Array
24
Narrow-Area Illumination For Working Range
Wide-Area Illumination Array -Near Field
25
Wide-Area Illumination For Near Field
Wide-Area Illumination Array -Far Field
26
Wide-Area Illumination For Far Field
CMOS Camera Board (34)
18
Input/ Output Subsystem
17
Multi-mode Image Processing Based Bar Code Symbol Reading Subystem
System Control Subsystem
19
A1
Image Capturing And Buffering Subsystem
16
Multi-mode Image Formation And Detection Subsystem
21
22
23
Field Of View Along Entire Working Range Of System
3-Tier Computing Platform (35,43)
Automatic IR-Based Object Presence And Range Detection Subsystem
20
Object Detection Field Overlapping Field Of View Of The System
12
Trigger Switch
2C
System-Mode Configuration Parameter Table
70

FIG. 2A1

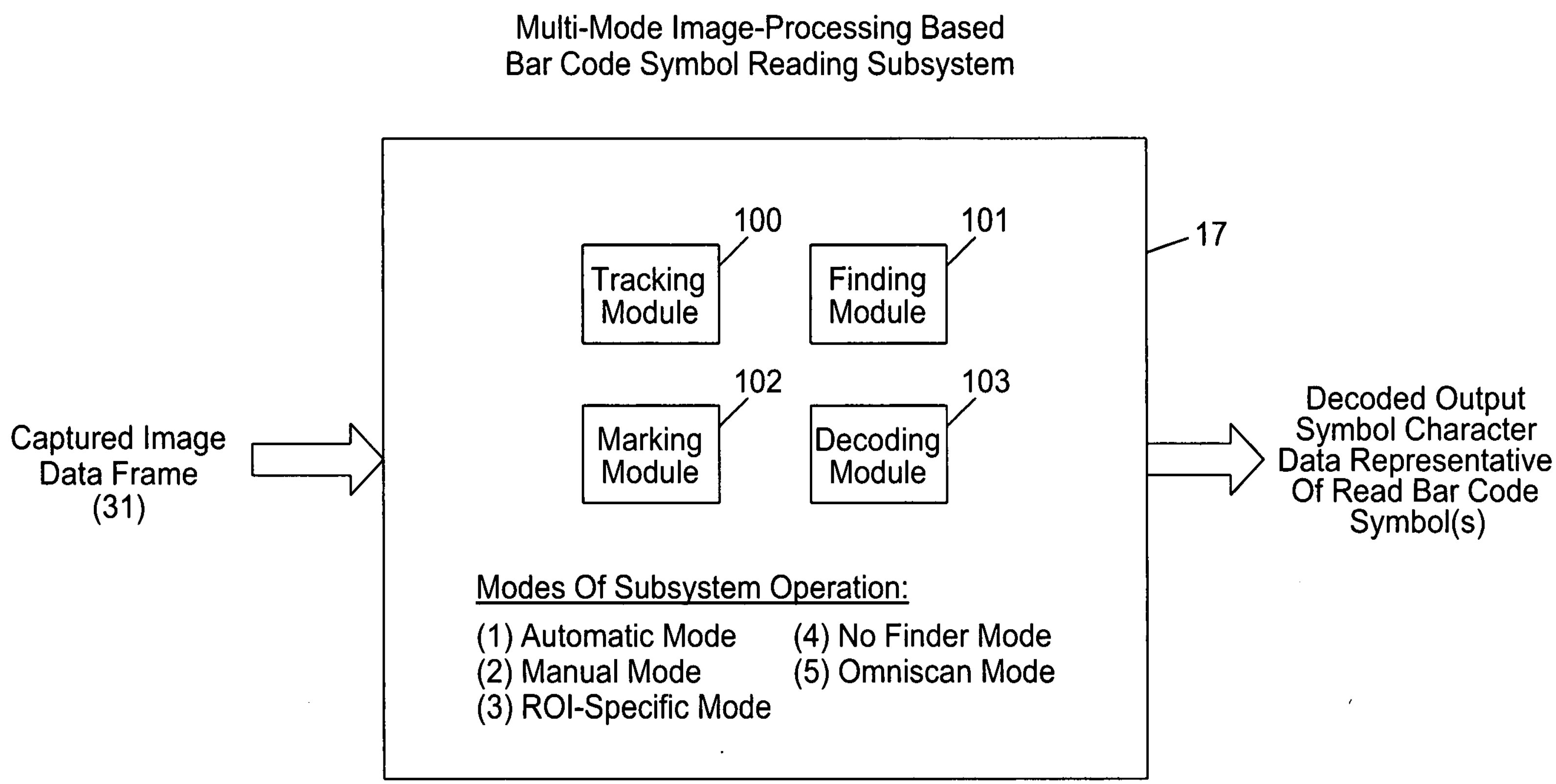

Multi-Mode Image-Processing Based
Bar Code Symbol Reading Subsystem
Captured Image
Data Frame
(31)
100
Tracking
Module
101
Finding
Module
102
Marking
Module
103
Decoding
Module
17
Modes Of Subsystem Operation:
(1) Automatic Mode
(2) Manual Mode
(3) ROI-Specific Mode
(4) No Finder Mode
(5) Omniscan Mode
Decoded Output
Symbol Character
Data Representative
Of Read Bar Code
Symbol(s)

FIG. 2A2

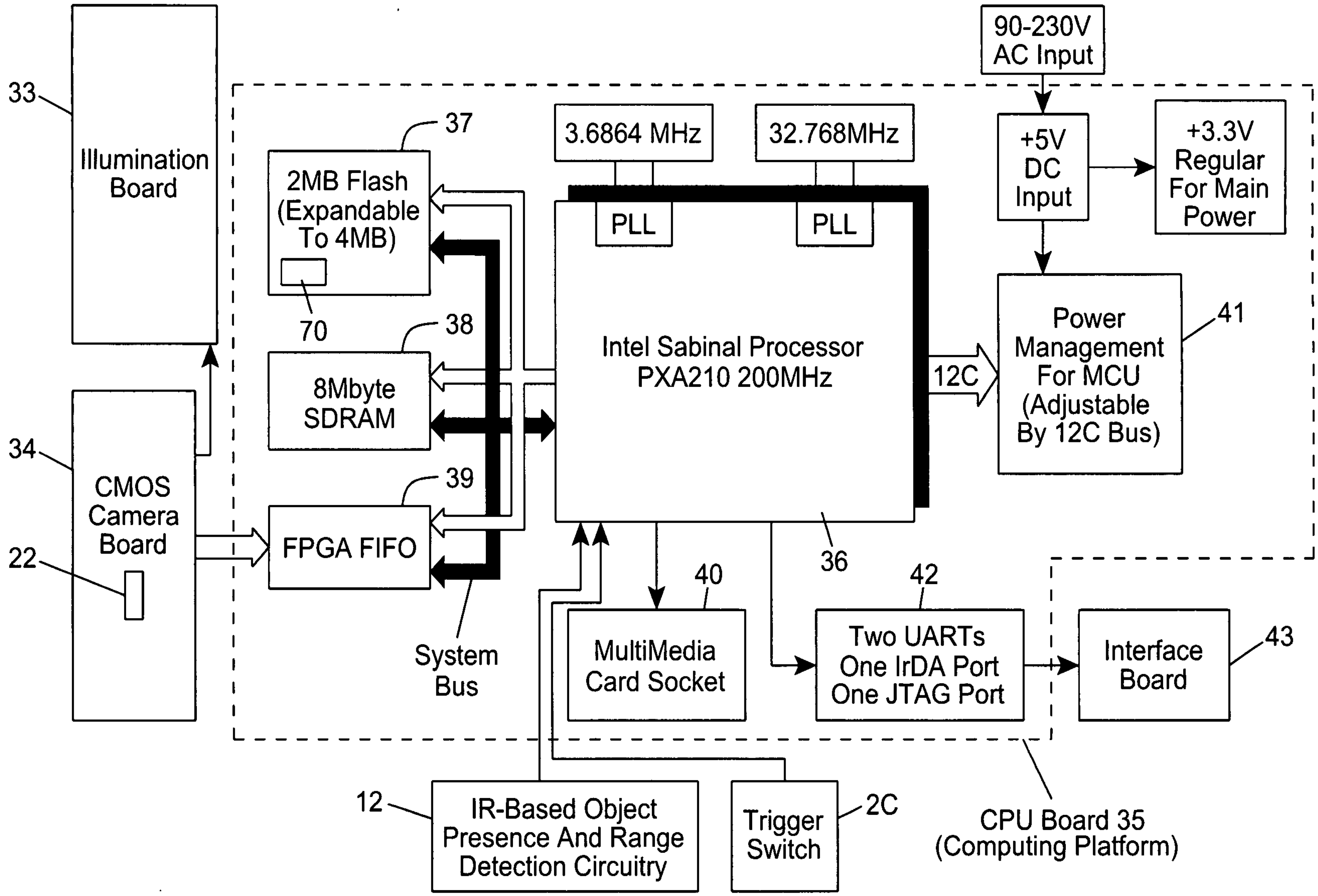

33
Illumination Board
34
CMOS Camera Board
22
2MB Flash (Expandable To 4MB)
37
70
8Mbyte SDRAM
38
FPGA FIFO
39
System Bus
3.6864 MHz
32.768MHz
PLL
PLL
Intel Sabinal Processor PXA210 200MHz
36
12C
90-230V AC Input
+5V DC Input
+3.3V Regular For Main Power
Power Management For MCU (Adjustable By 12C Bus)
41
MultiMedia Card Socket
40
Two UARTs One IrDA Port One JTAG Port
42
Interface Board
43
12
IR-Based Object Presence And Range Detection Circuitry
Trigger Switch
2C
CPU Board 35 (Computing Platform)

FIG. 2B

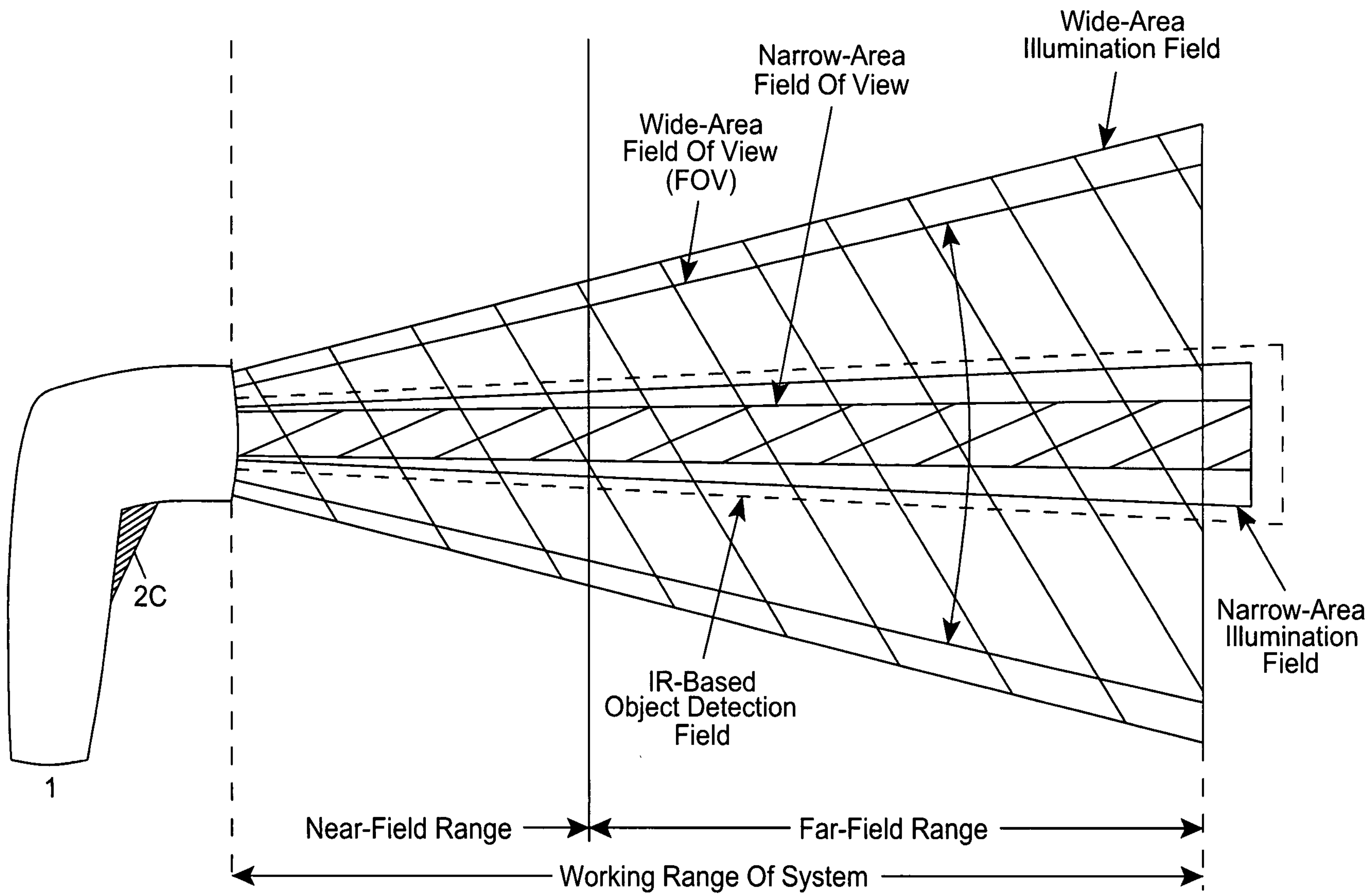

Wide-Area
Illumination Field
Narrow-Area
Field Of View
Wide-Area
Field Of View
(FOV)
Narrow-Area
Illumination
Field
IR-Based
Object Detection
Field
2C
1
Near-Field Range
Far-Field Range
Working Range Of System

FIG. 3A

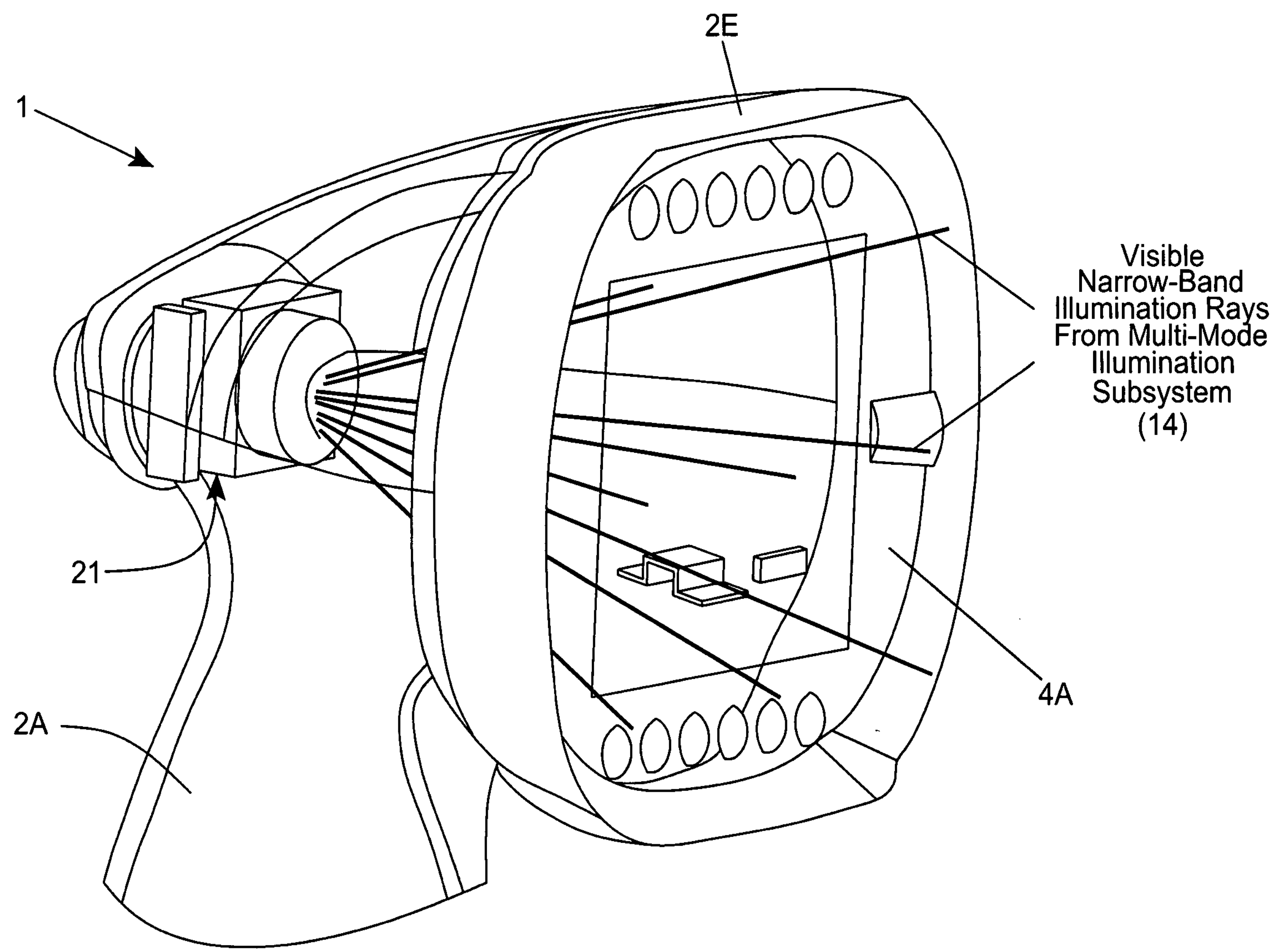

1
2E
Visible
Narrow-Band
Illumination Rays
From Multi-Mode
Illumination
Subsystem
(14)
21
2A
4A

FIG. 3B

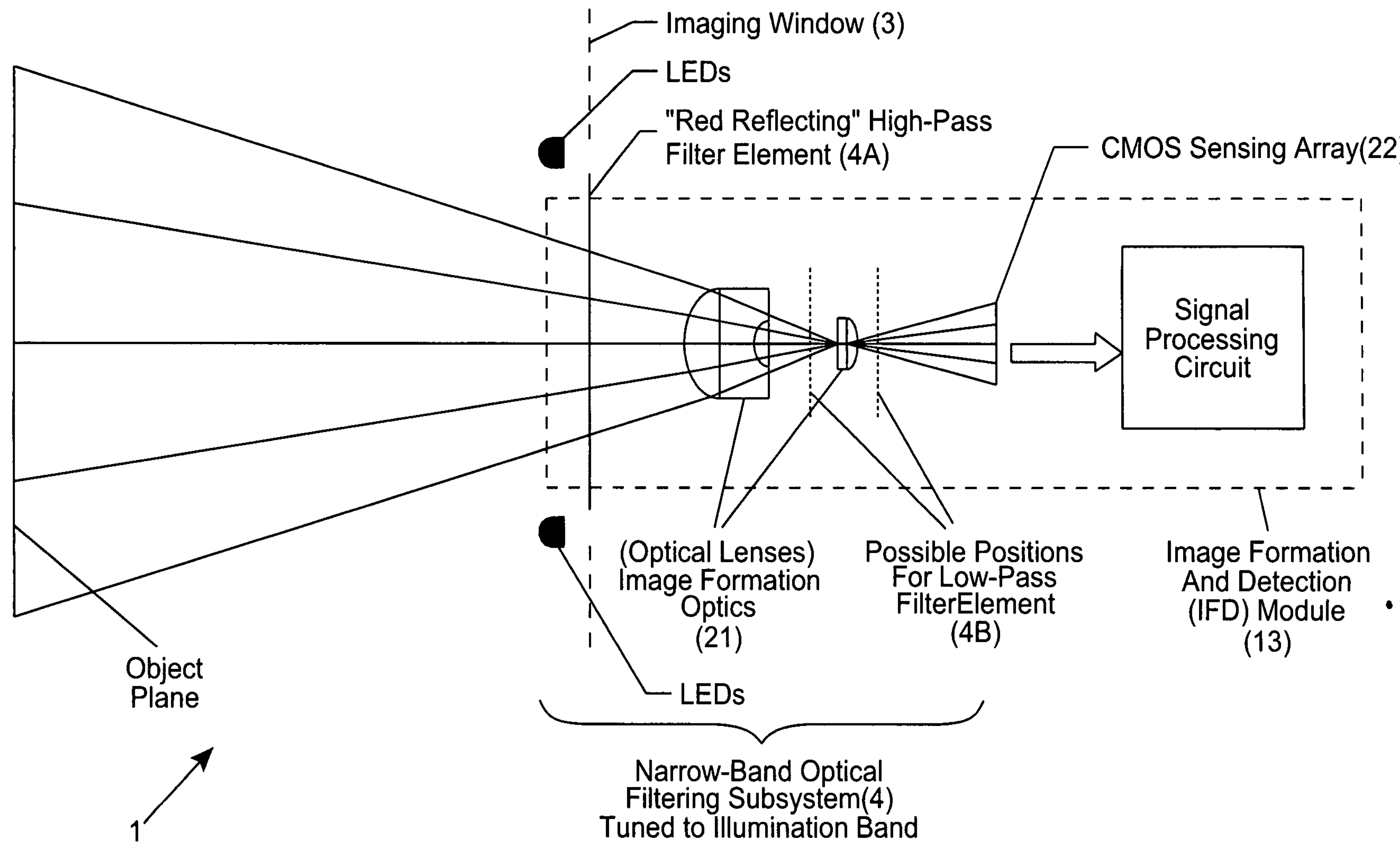

Imaging Window (3)
LEDs
"Red Reflecting" High-Pass
Filter Element (4A)
CMOS Sensing Array(22)
Signal
Processing
Circuit
(Optical Lenses)
Image Formation
Optics
(21)
Possible Positions
For Low-Pass
FilterElement
(4B)
Image Formation
And Detection
(IFD) Module
(13)
Object
Plane
LEDs
Narrow-Band Optical
Filtering Subsystem(4)
Tuned to Illumination Band
1

FIG. 3C

- 45° FOV √
- As Few Elements As Possible √
  - Previous Designs Had 4 Or 5
- As Small As Possible √
  - Max Diameter = 12mm
- All Spherical Surfaces √
- Common Glasses √
  - LaK2 (≈LaK9)
  - ZF10 (=SF8)
  - LaF2 (≈LaF3)

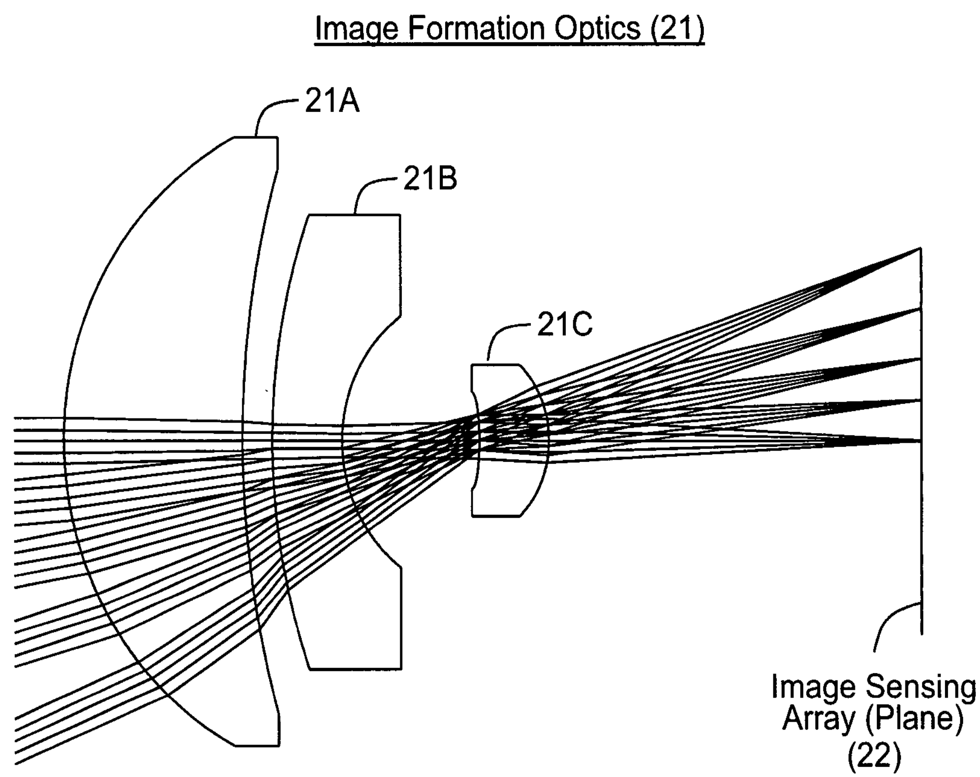


FIG. 3D

- Barrel Hold Lens Elements
- Base Hold Sensors
- Barrel Slides In Base To Focus

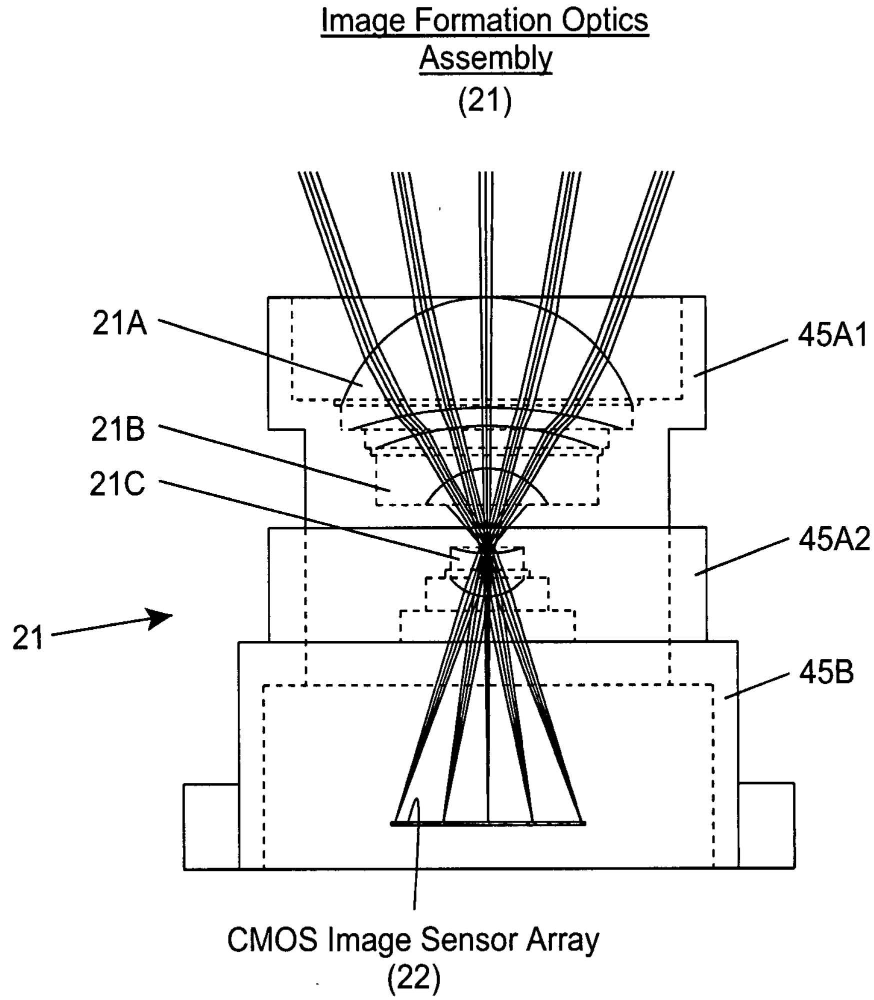


FIG. 3E

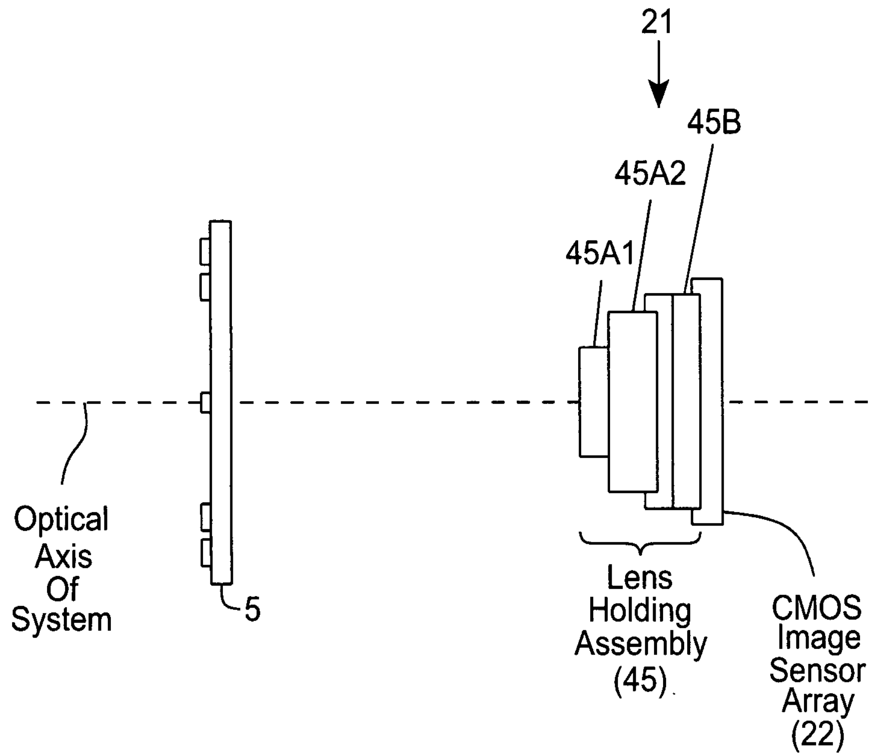


FIG. 3F1

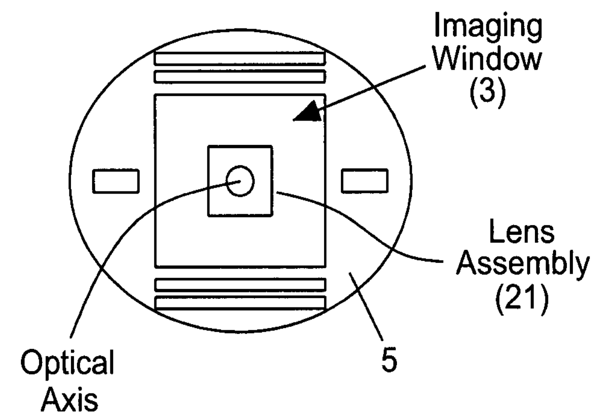
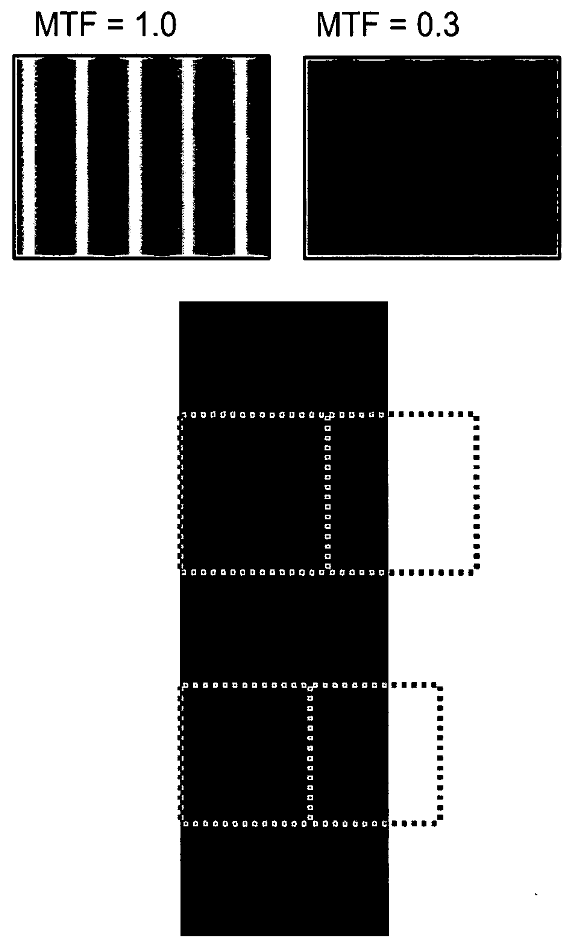


FIG. 3F2

DOF Determination Of Image Formation Optics

- At each distance, find frequency where MTF drops to 0.3
  - Rule of thumb for bar code decoding
  - Depends on code, speed, etc, etc - must test



- BUT: limited by sampling requirement
  - Software needs ~1.6 pixels on narrow code element
  - Limits decode ability regardless of optics
  - Exact value is rule of thumb and flexible (1.4 – 1.6)

FIG. 3G

Depth Of Field

- Face To 8" For 13.5 Mil
- Optics Resolve 4 Mil Somewhere
- Decodes 5 Mil Somewhere
- No Moving Elements

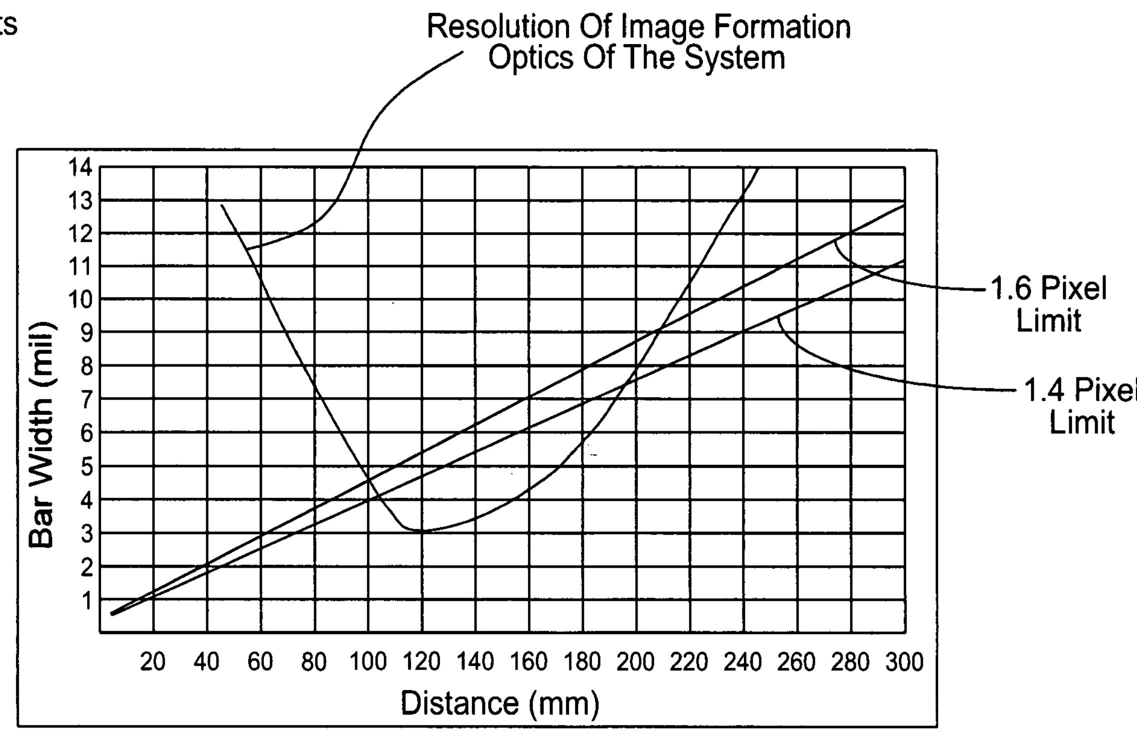


FIG. 4A

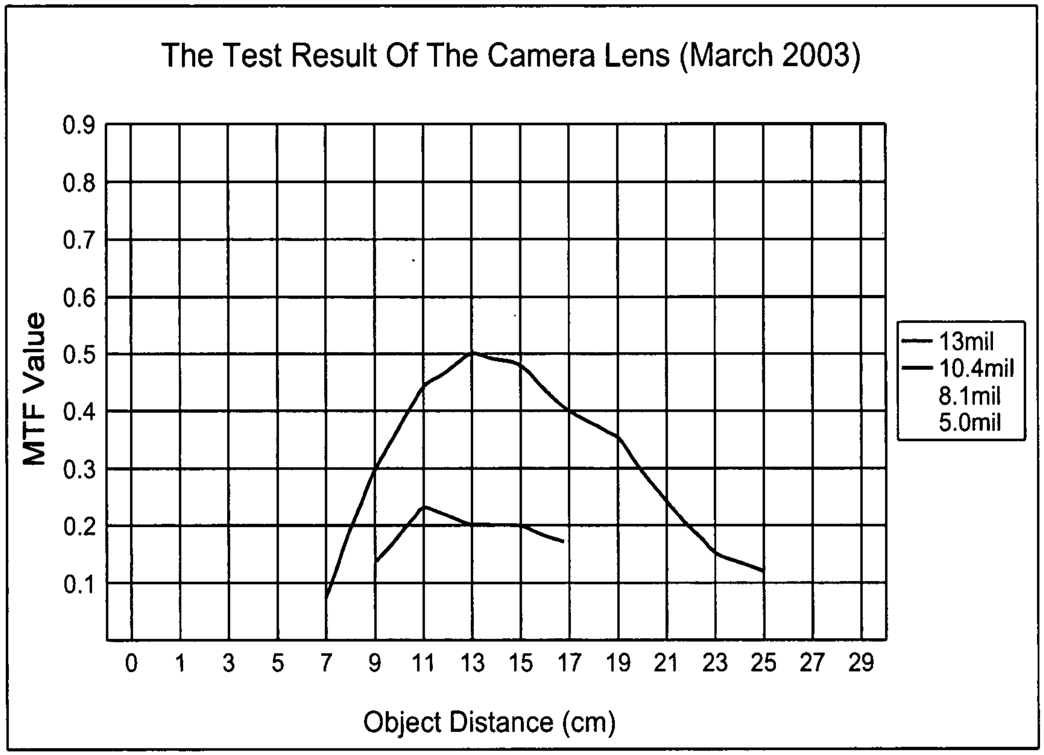

The Test Result Of The Camera Lens (March 2003)
MTF Value
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
Object Distance (cm)
13mil
10.4mil
8.1mil
5.0mil

FIG. 4B

Depth Of Field

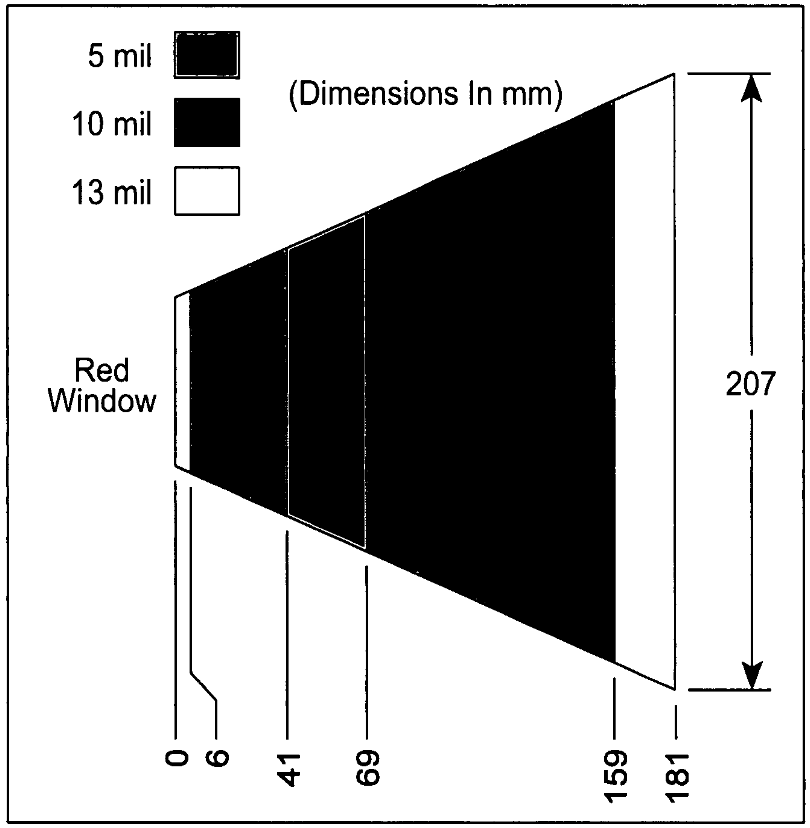
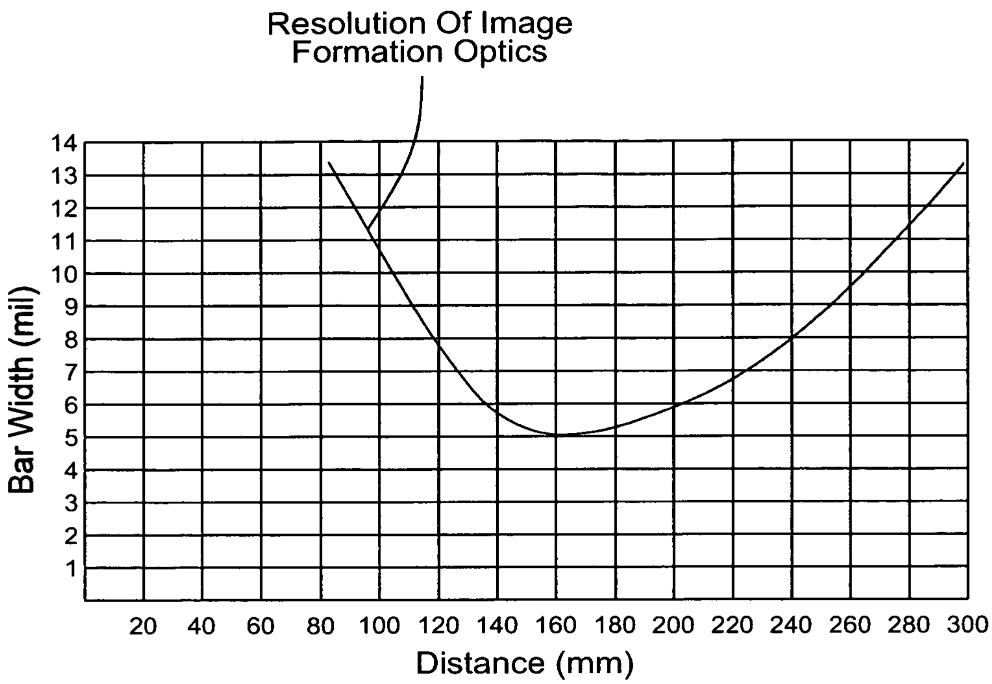
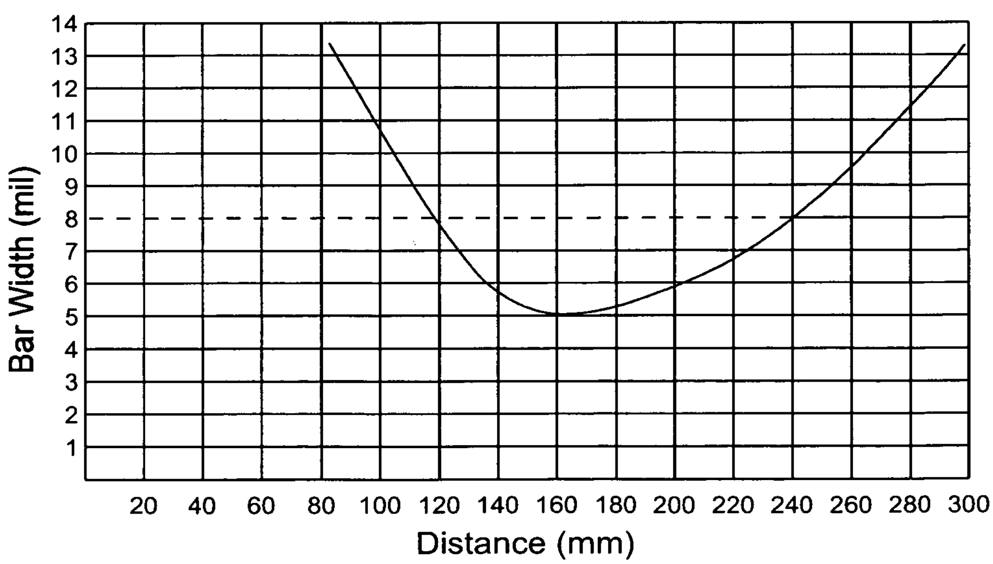


FIG. 4C



Resolution Of Image Formation Optics
14
13
12
11
10
9
8
7
6
5
4
3
2
1
Bar Width (mil)
20 40 60 80 100 120 140 160 180 200 220 240 260 280 300
Distance (mm)

FIG. 4D



14
13
12
11
10
9
8
7
6
5
4
3
2
1
Bar Width (mil)
20 40 60 80 100 120 140 160 180 200 220 240 260 280 300
Distance (mm)

FIG. 4E

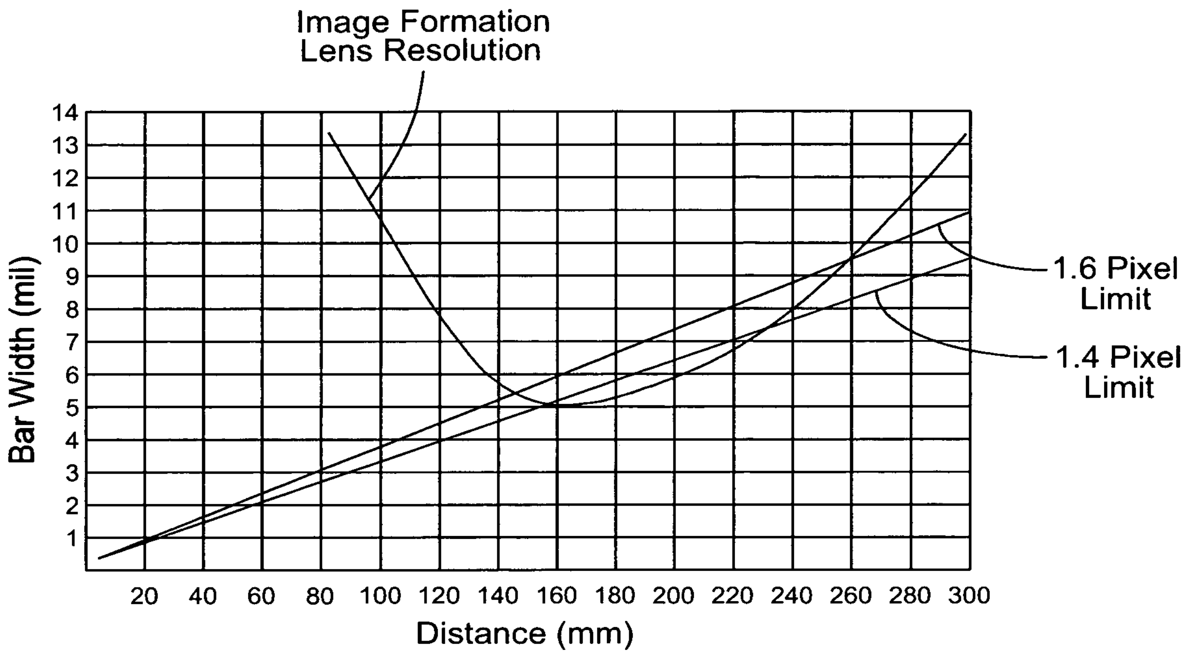
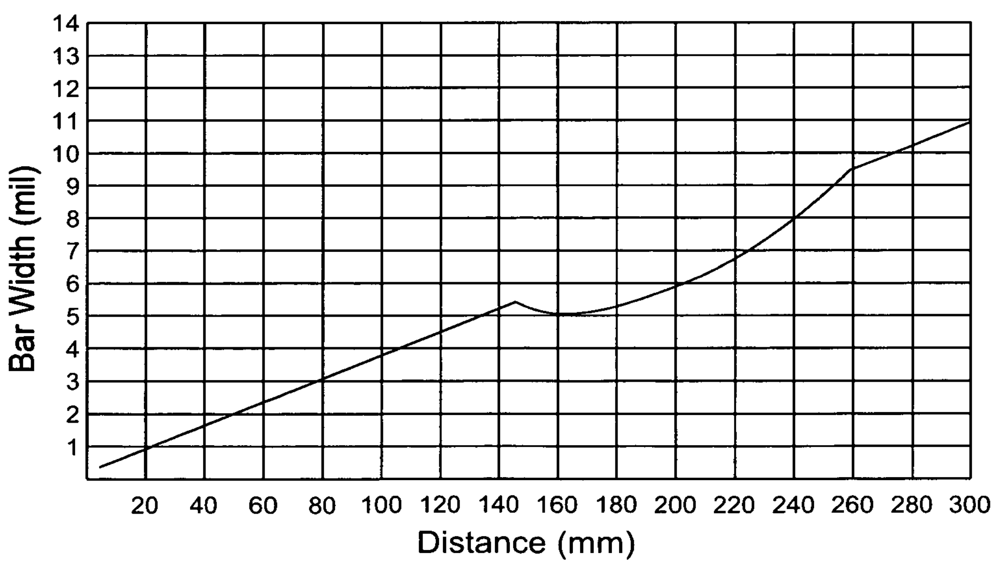


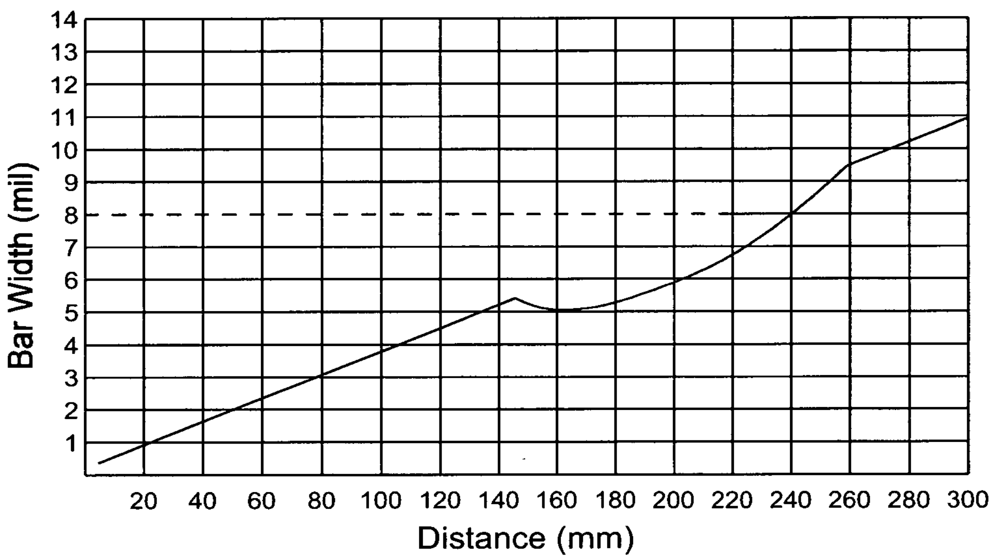
Image Formation
Lens Resolution
1.6 Pixel
Limit
1.4 Pixel
Limit
Bar Width (mil)
14
13
12
11
10
9
8
7
6
5
4
3
2
1
20 40 60 80 100 120 140 160 180 200 220 240 260 280 300
Distance (mm)

FIG. 4F



Bar Width (mil)
14
13
12
11
10
9
8
7
6
5
4
3
2
1
20 40 60 80 100 120 140 160 180 200 220 240 260 280 300
Distance (mm)

FIG. 4G



Bar Width (mil)
14
13
12
11
10
9
8
7
6
5
4
3
2
1
20 40 60 80 100 120 140 160 180 200 220 240 260 280 300
Distance (mm)

FIG. 4H

DOF_PMAG.zpl

```
graphics
xmx=xmax()
xmn=xmin()
ymx=ymax()
 ymn=ymin()
xwidth=xmx-xmn
ywidth=ymx-ymn
xleft=xmn+(0.l*xwidth)
xrigh=xmn+(0.95*xwidth)
ytopp=ymn+(0.05*ywidth)
ybott=ymn+(0.7*ywidth)

line xleft,ytopp,xrigh,ytopp
line xrigh,ytopp,xrigh,ybott
line xrigh,ybott,xleft,ybott
line xleft,ybott,xleft,ytopp

format 4.3
settextsize 140,80
gtext 0.68*xwidth,(0.85)*ywidth,0,"Wav : "
gtext 0.68*xwidth, (0.88)*ywidth,0, "WGT : "
for i=l,nwav(),1
gtext (0.68+i*0.05)*xwidth,0.85*ywidth,0,$str(wavl(i))
gtext (0.68+i*0.05)*xwidth,0.88*ywidth,0,$str(wwgt(i))
next
gtext 0.68*xwidth,(0.91)*ywidth,0,"Relative illumination: "
gtext 0.9*xwidth,(0.91)*ywidth,0,$str(reli(nfld()))
settextsize 90,50
input "Please input startpoint (mm):",start
if (start<=0) then input "Please input startpoint (mm):", start
input "Please input pixel size (um):",pix
if (pix<=O) then input "Please input pixel size (um):",pix
for i=start,start+150,10
xpos=xleft+(i-start)/150*0.85*xwidth
line xpos,ytopp,xpos,ybott
format3.0
gtext xleft*0.85+(i-start)/150*0.85*xwidth,0.72*ywidth,0,$str(i)
next
settextsize 70,40
for i=1,14,1
ypos=ytopp+i/14*.65*ywidth
line xleft,ypos,xrigh,ypos
format 3.0
gtext 0.05*xwidth,ytopp*0.9+(j-1)/14*.65*ywidth,0,$str(14-i+1)
next

gtitle "The DOF and PMAG curve of current design"
gdate

format 12.6
oldthic=thic(0)

getsystemdata 2
settextsize 120,40
j=1
gtext xwidth*0.018,0.85*ywidth,0,"centering "
for i=1,nsur()-2,1
        if (gind(i)!=0.0)
                format 2.0
                gtext xwidth*0.10+(j-1)*0.07*xwidth,0.85*ywidth ,0,$str(j)+":"
                gtext xwidth*0.12+(j-1)*0.07*xwidth,0.85*ywidth ,0,":"
                format 4.2
```

FIG. 4I1

```
                    if(curv(i)*curv(i+1)<0) then
centering=abso((sdia(i)*curv(i)+sdia(i+1)*curv(i+1)))
                    if(curv(i)*curv(i+1)>0) then
centering=abso((sdia(i)*curv(i)-sdia(i+1)*curv(i+1)))
                    gtext xwidth*0.13+(j-1)*0.07*xwidth,0.85ywidth,0,$str(centering)
                    j=j+1
        endif
next
format 4.2
settextsize 70,40
gtext xwidth*0.018,0.91*ywidth,0,"image space f/# : "+$str(vec2(8))
gtext xwidth*0.018,0.94*ywidth,0,"effective focal length: "+ $str(vec2(7))
!color (3)
gtextcent ymn+(0.77*ywidth),"distance (mm)"
gtext xleft*0.32,0.5*ywidth,90,"bar width (mil)"

format 12.6
settextsize 100,40
minmtf=1
maxfreq=0
thic 0=start
update all
for k=0,200,0.2
          !i=nfld()
           for i=1,nfld(),1
                     getmtf k,O,i,2,1,1
                     !print vec1(0)
                     !print vec1(1)
                     if (vec1(0)<minmtf) then minmtf=vec1(0)
                     if (vec1(1)<minmtf) then minmtf=vec1(1)
                     if (minmtf<=0.3)
                               maxfreq=k
                               goto 1
                     endif
           next
next
label 1
!color (1)

!output "1.txt" append

oldxpos=xleft+0/150*0.85*xwidth
oldypos=ytopp+(14-(1/(maxfreq/(sdia(0)/sdia(nsur())))*0.5/25.4*1000))/14*0.65*ywidth
switch=0
m=0
for j=start,start+150,3
           thic 0=j
          update all
           minmtf=1
           for k=m,200,0.3
                     !i=nfld()
                     for i=1,nfld(),1
                     getmtf k,0,i,2,1,1
                     if (vec1(0)<minmtf) then minmtf=vec1(0)
                     if (vec1(1)<minmtf) then minmtf=vec1(1)
                     if (minmtf<=0.3)
                               maxfreq=k
                               goto 2
                      endif
                      next
           next
           label 2
           if (maxfreq-5)>0
```

FIG. 4I2

```
            rn=maxfreq-10
    else
            rn=0
    endif
    !print j,sdia(0),sdia(nsur()),maxfreq
    if ((switch==0) & (1/(maxfreq/(sdia(0)/sdia(nsur()))) 0.5/25.4*1000<=13))
          !color (0)
          format 5.2
          a$="FOV for 10 mil: "+$str(2*sdia(0)) + " at "+$str(j-2)+ mm ; "
          gtext xwidth*0.018,0.97*ywidth,0,a$
          switch=1
          format 12.6
          !color(1)
    else
                  if ((switch==1) &
(1/(maxfreq/(sdia(0)/sdia(nsur())))*0.5/25.4*1000>=13))
                        !color(0)
                        format 5.2
                        a$=$str(2*sdia(0))+" at "+$str(j-2)+" mm"
                        gtext xwidth*0.44,0.97*ywidth,0,a$
                        switch=0
                        format 12.6
                        goto 3
                        !color(1)
                  endif
          endif
          newxpos=        xleft+(j-start)/150*0.85*xwidth

newypos=ytopp+(14-(1/(maxfreq/(sdia(0)/sdia(nsur())))*0.5/25.4*1000))/14*0.65*ywidth
              if ((14-14*(oldypos-ytopp)/0.65/ywidth)<14) then line
oldxpos,oldypos,newxpos,newypos
              oldxpos=newxpos
              oldypos=newypos
next
label 3
thic 0=start
update all
oldxpos=xleft+0/150*0.85*xwidth
oldxpos1=xleft+0/150*0.85*xwidth
oldypos=ytopp+(14-(0.5/((0.5/1.6/pix*1000)/(sdia(0)/sdia(nsur())))/25.4*1000))/14*0.
65*ywidth
oldypos1=ytopp+(14-(0.5/((0.5/1.4/pix*1000)/(sdia(0)/sdia(nsur())))/25.4*1000))/14*0
.65*ywidth
for j=start,start+150,4
              thic 0=j
              update all
              newxpos=xleft+(j-start)/150*0.85*xwidth
              newxpos1=xleft+(j-start)/150*0.85*xwidth

newypos=ytopp+(14-(0.5/((0.5/1.6/pix*1000)/(sdia(0)/sdia(nsur())))/25.4*l000))/14*0.
65*ywidth

newypos1=ytopp+(14-(0.5/((0.5/1.4/pix*1000)/(sdia(0)/sdia(nsur())))/25.4*1000))/14*0
.65*ywidth
            line oldxpos, oldypos,newxpos, newypos
            line oldxpos1,oldypos1,newxpos1,newypos1
            oldxpos=newxpos
            oldypos=newypos
            oldxpos1=newxpos1
            oldypos1=newypos1
next

thic 0=oldthic
```

FIG. 4I3

Multi-Mode Illumination Subsystem

- Three Modes Of Illumination
  - (1) Wide-Area For "Near" Object (0 mm-100 mm)
  - (2) Wide-Area For "Far" Object (100 mm-200 mm)
  - (3) Narrow-Area For "Near" Object (30 mm-100 mm)

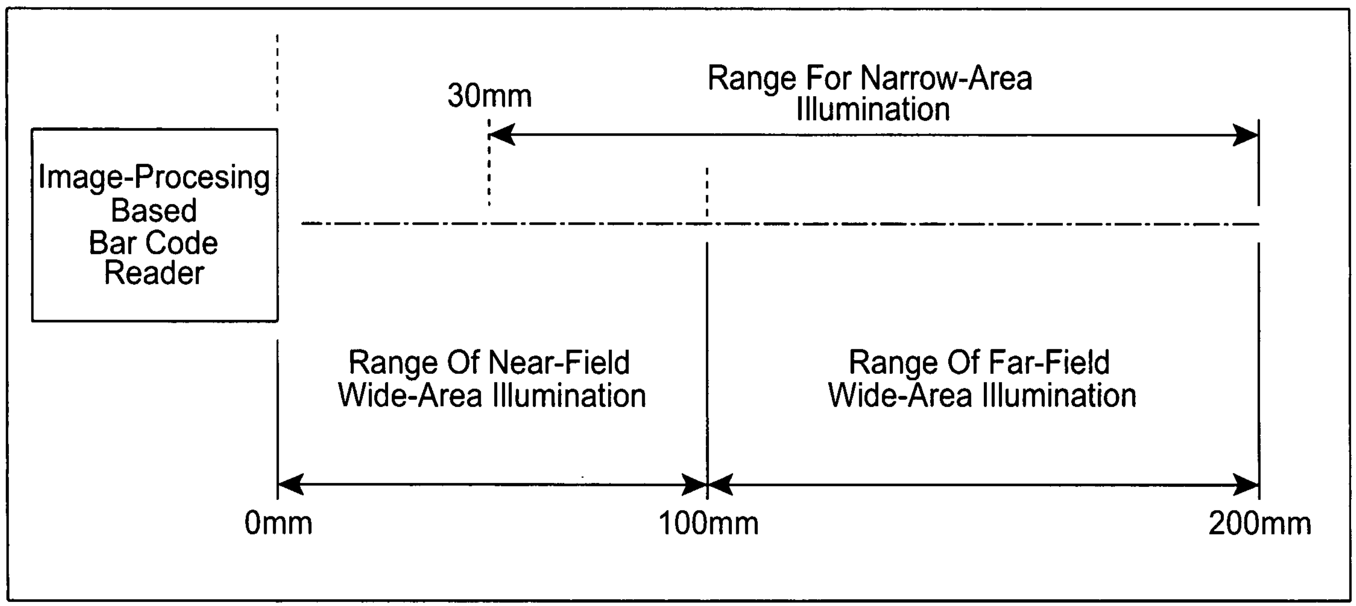


FIG. 5A1

Illumination Design Goals For First Illustrative Embodiment

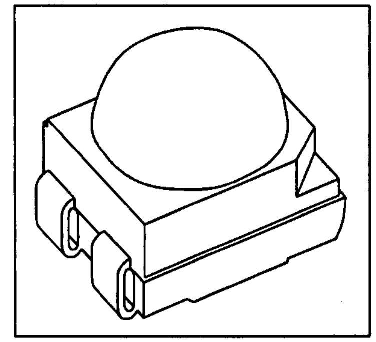
- Wide-Area Illumination Modes
  - Match FOV and DOF (45°, 200mm)
  - Sufficient power density on target
    - Pixel value > 80 DN at far field center
  - Achieve sufficient uniformity (center:edge = 2:1 max)
  - Use as few LEDs as possible

- Narrow-Area Illumination Mode
  - Line usable beginning 40 mm from window
  - Match FOV and DOF
  - Sufficient power density on target
  - Sufficiently thin line
    - Height < 10 mm at far field

FIG. 5A2

LEDs For Narrow-Area Illumination

- Linear Illumination: Osram LS E655
  - 633 nm InGaAlP
  - 60° Lambertian Emittance
  - 6.75 mW Total Output Power (Typical Conditions)
  - $0.18 Each In 50k



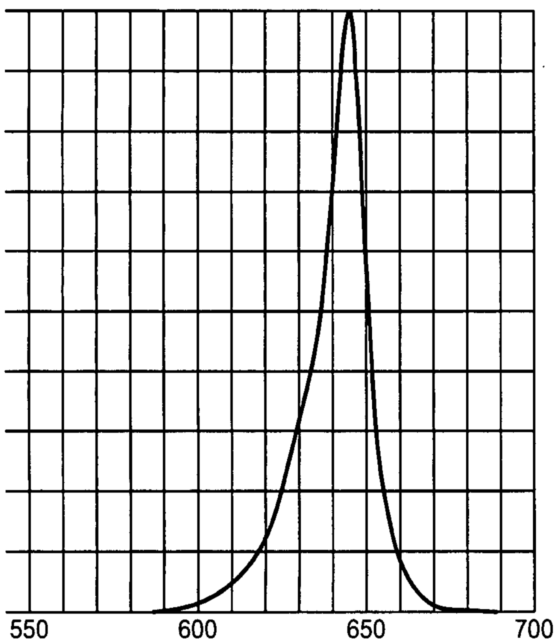


FIG. 5C1

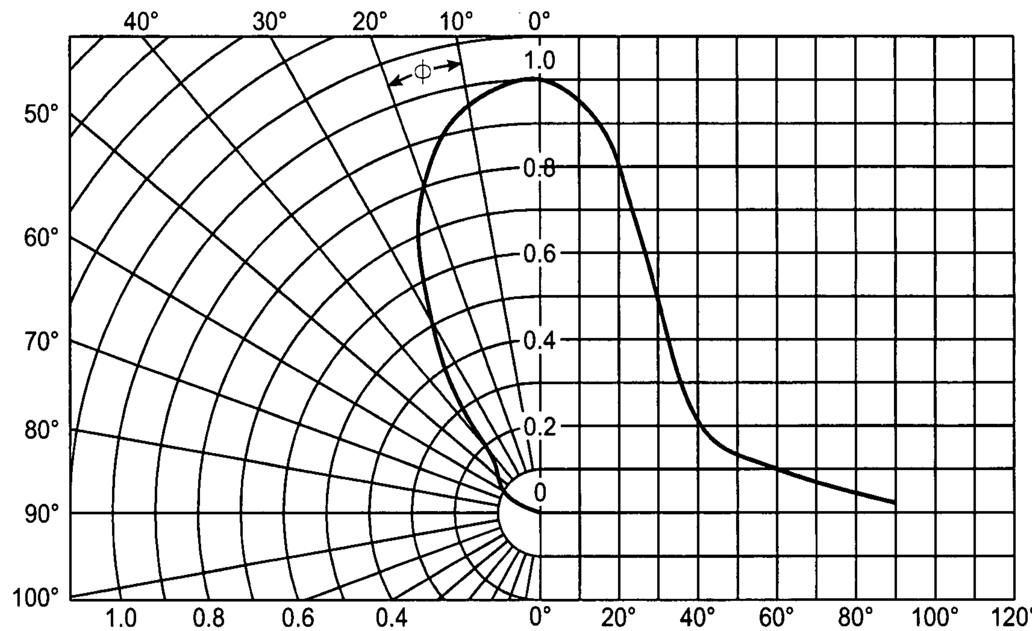


FIG. 5C2

LED Arrangements For Near-Field And Far-Field Wide Area Illumination Arrays And Narrow-Area Illumination Arrays

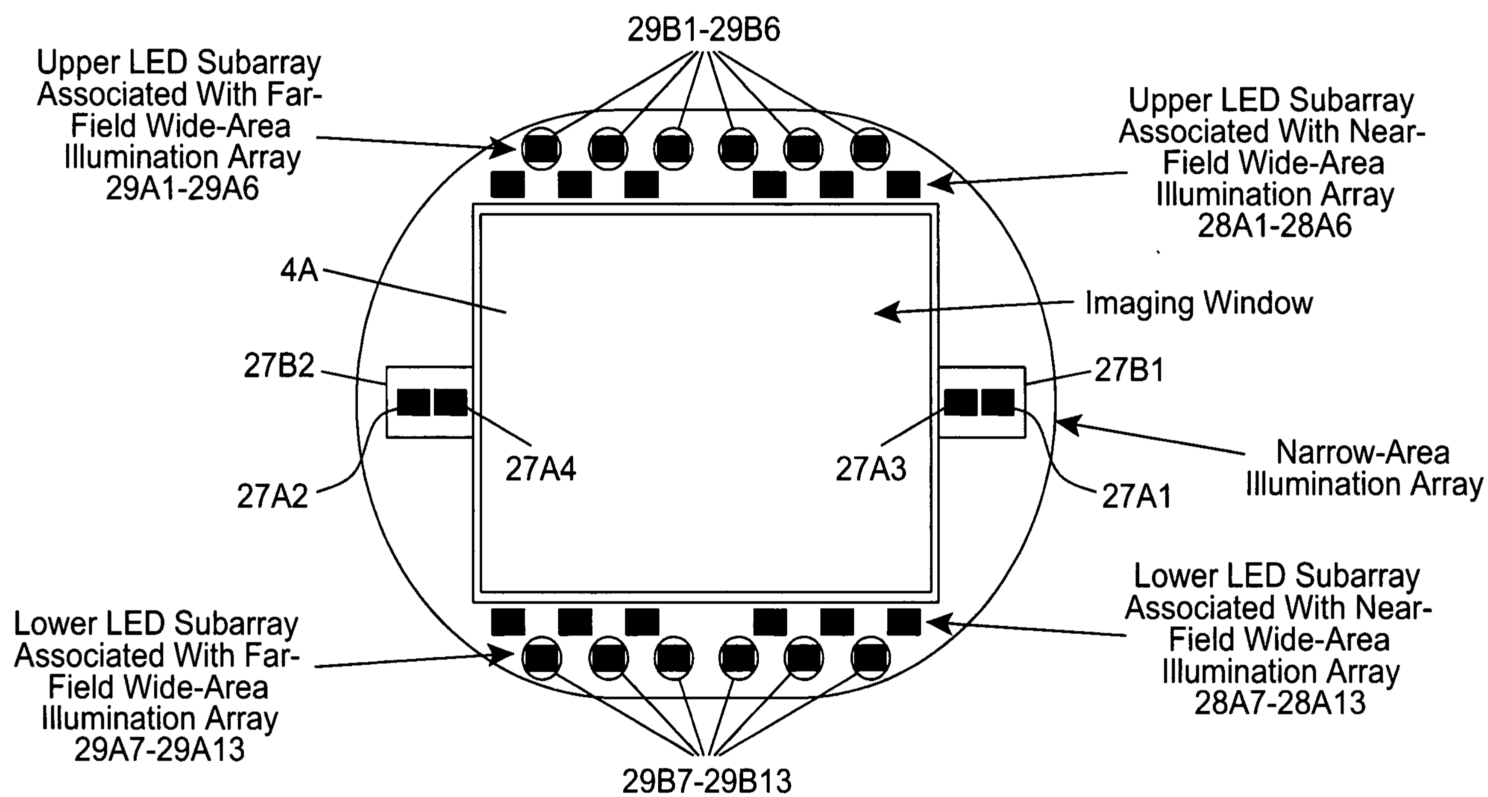


FIG. 5B

Cylindrical Lenses For Narrow-Area Illumination Array

- First Surface Curved Vertically To Create Line
- Second Surface Curved Horizontally To Control Line Height

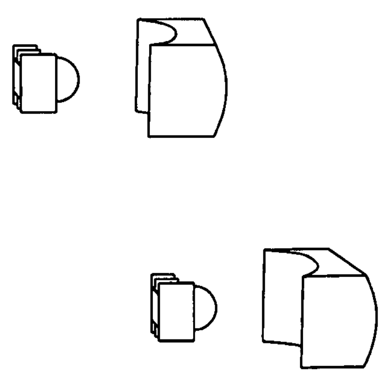


FIG. 5C3

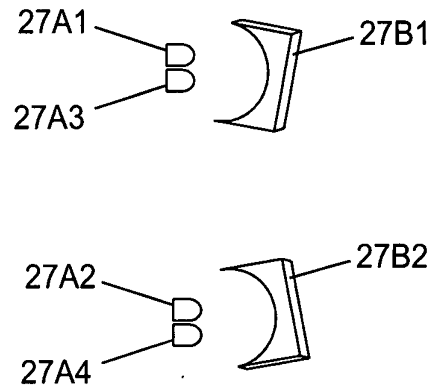


FIG. 5C4

Linear Illumination Profiles

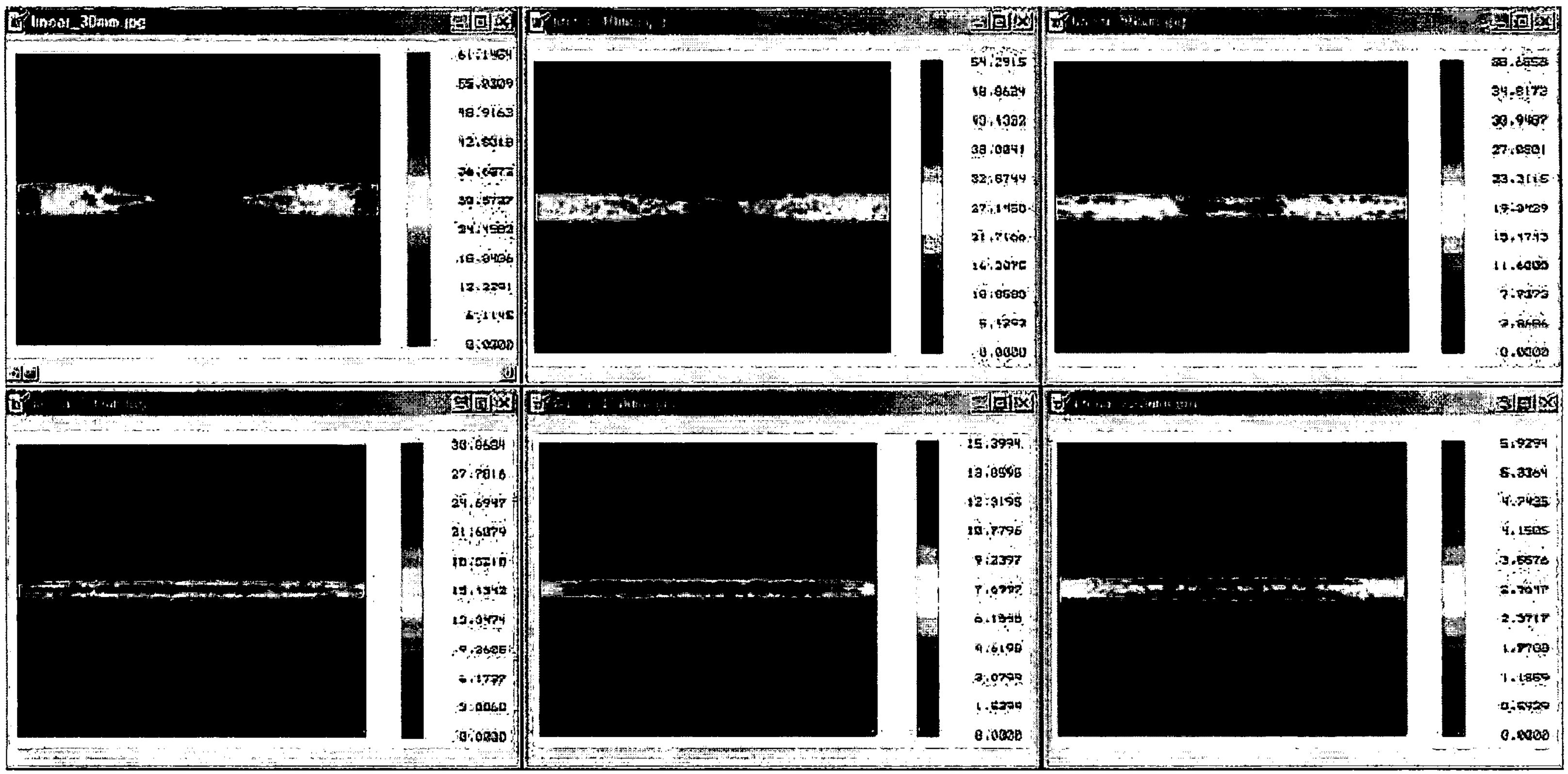
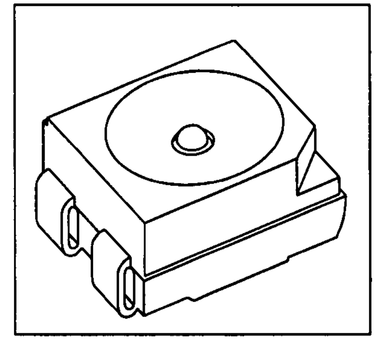


FIG. 5C5

Area LEDs

- Area Illumination: Osram LS E67B
  - 633 nm InGaAlP
  - 120° Lambertian Emittance
  - 11.7 mW Total Output Power (Typical Conditions)
  - $0.18 each In 50k



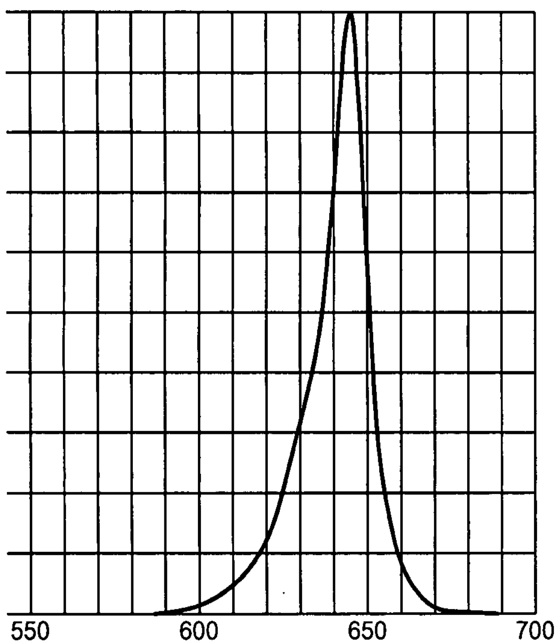


FIG. 5D1

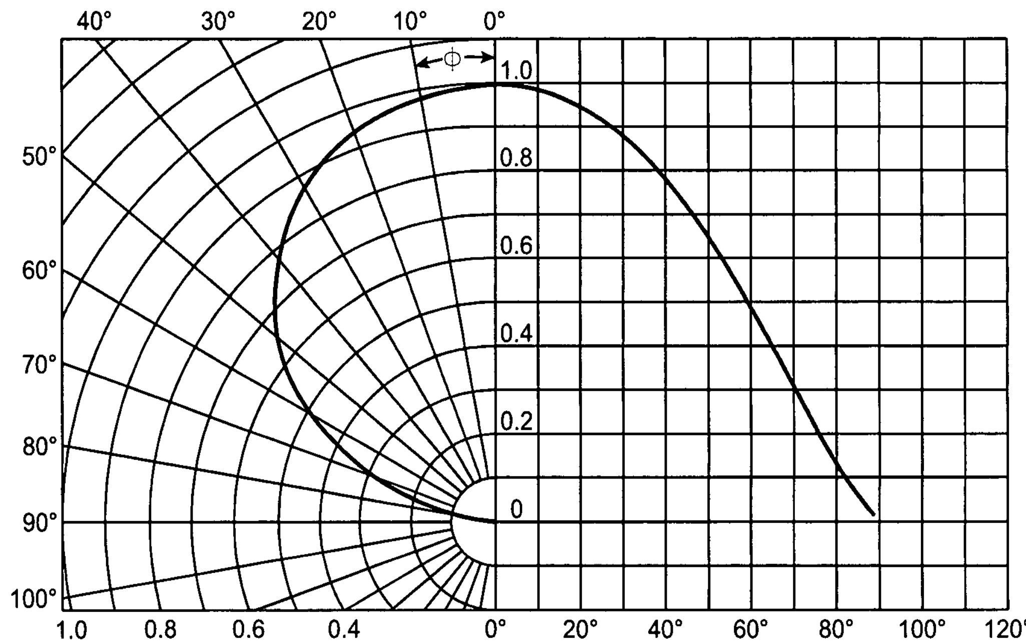


FIG. 5D2

Far Area Lenses

- Plano Convex Lenses In Front Of Far Field LEDs

- Light Aimed By Angling Lenses
  - Even Out Distribution Across FOV Throughout DOF
  - Satisfy Center: Edge = 2:1 Max Criterion
  - Allows LEDs To Be Mounted Flat

- All Lenses CNCed In Single Piece Of Plastic

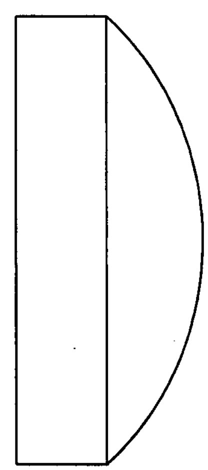


FIG. 5D3

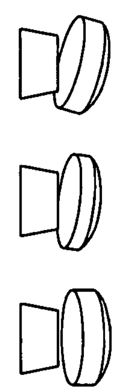


FIG. 5D4

Wide-Area Illumination Profiles (Near)

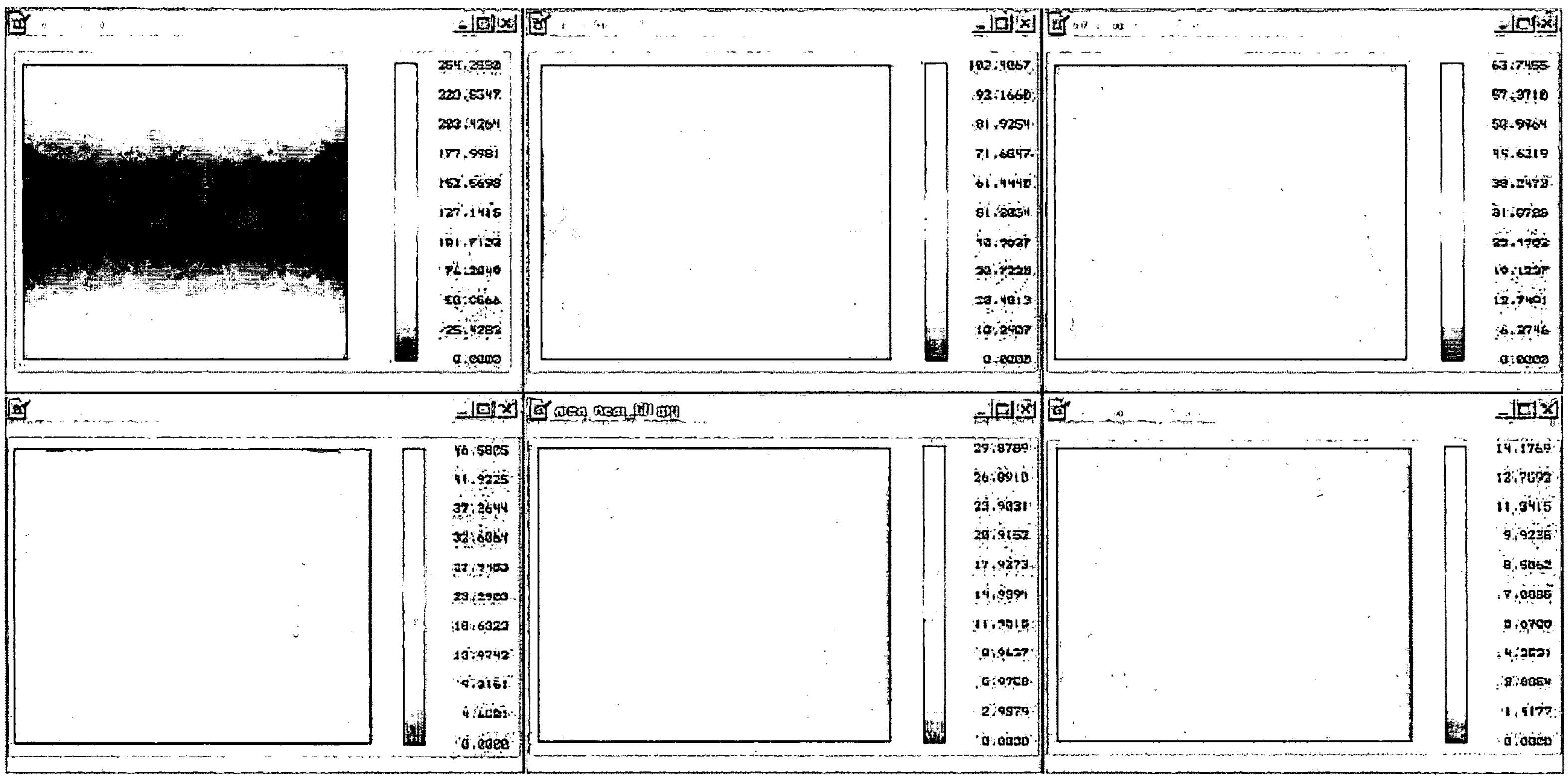


FIG. 5D5

Wide-Area Illumination Profiles (Far)

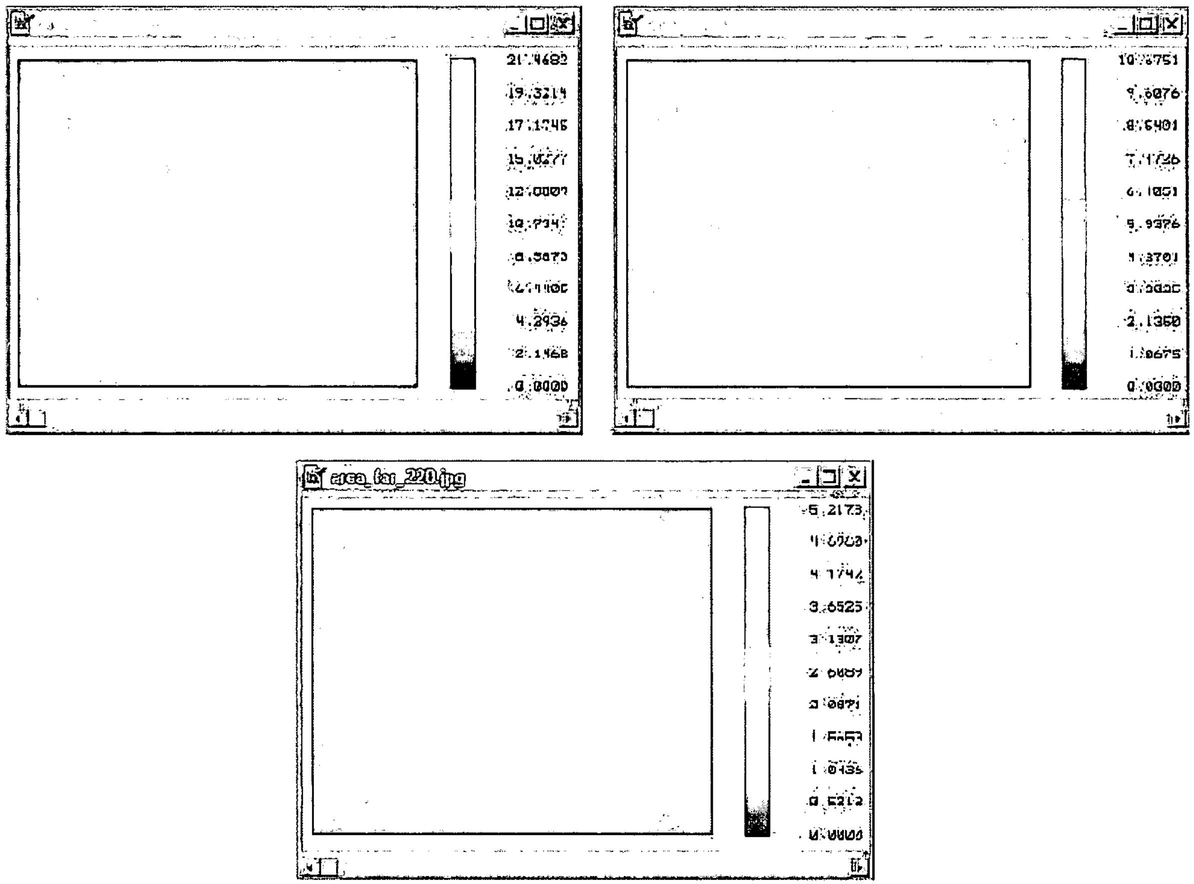


FIG. 5D6

Pixel Value Calculation

- Pixel Value Calculation For Center Of Far-Field Shows Sufficient Signal (> 80DN)

Description		Value	Unit
Sensor Power Density	Target Power Density	4	µW/mm²
	Surface Reflection	0.6	#
	Optical Transmittance	0.9	
	F-Number	9	
	Pixel Power Density	0.007	µW/mm²
Signal	CMOS Internal Gain	4.5	#
	Amplification Gain	20	dB
	Integration Gain	5	ms
	Sensor Responsivity	1.8	V / (lx s)
	Wavelength	633	nm
	Photopic Luminous Efficiency	0.238	lm / W
	Signal Out Of Sensor	0.439	V
Value Pixel	A/D Range Max	1.3	V
	A/D Range Min	0.0	
	Pixel Value (0-255)	86	DN

FIG. 5D7

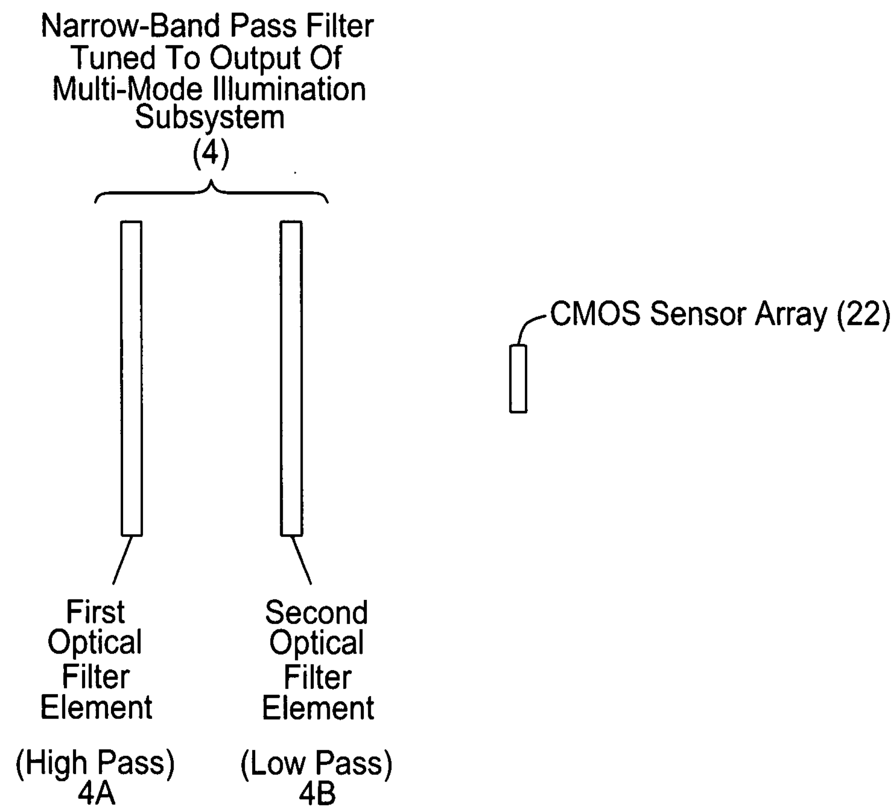

Narrow-Band Pass Filter Tuned To Output Of Multi-Mode Illumination Subsystem (4)
CMOS Sensor Array (22)
First Optical Filter Element
(High Pass) 4A
Second Optical Filter Element
(Low Pass) 4B

FIG. 6A1

Red Window And Low-Pass Filter Characteristics

- Must Bandpass Return Light Against Ambient
  - Red Window + Low Pass Filter
  - Restricts Range To 620nm – 700nm

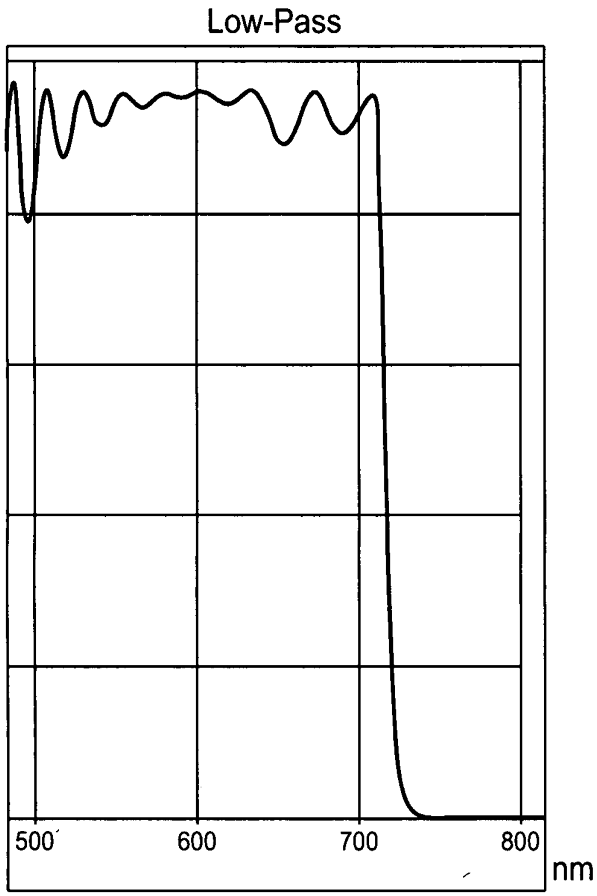


FIG. 6A2

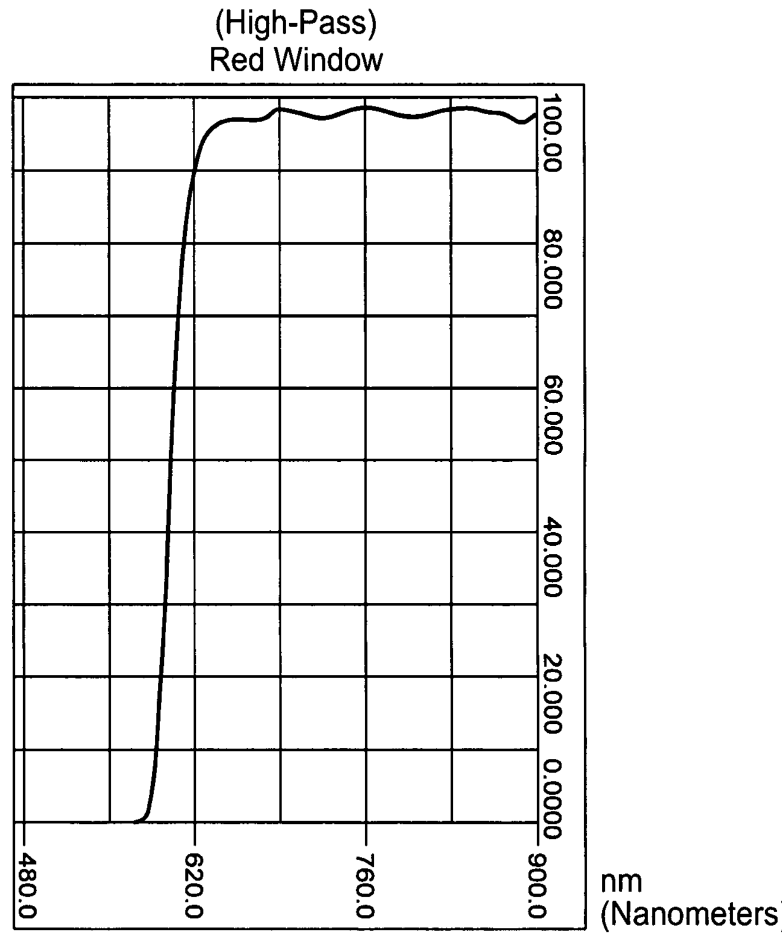
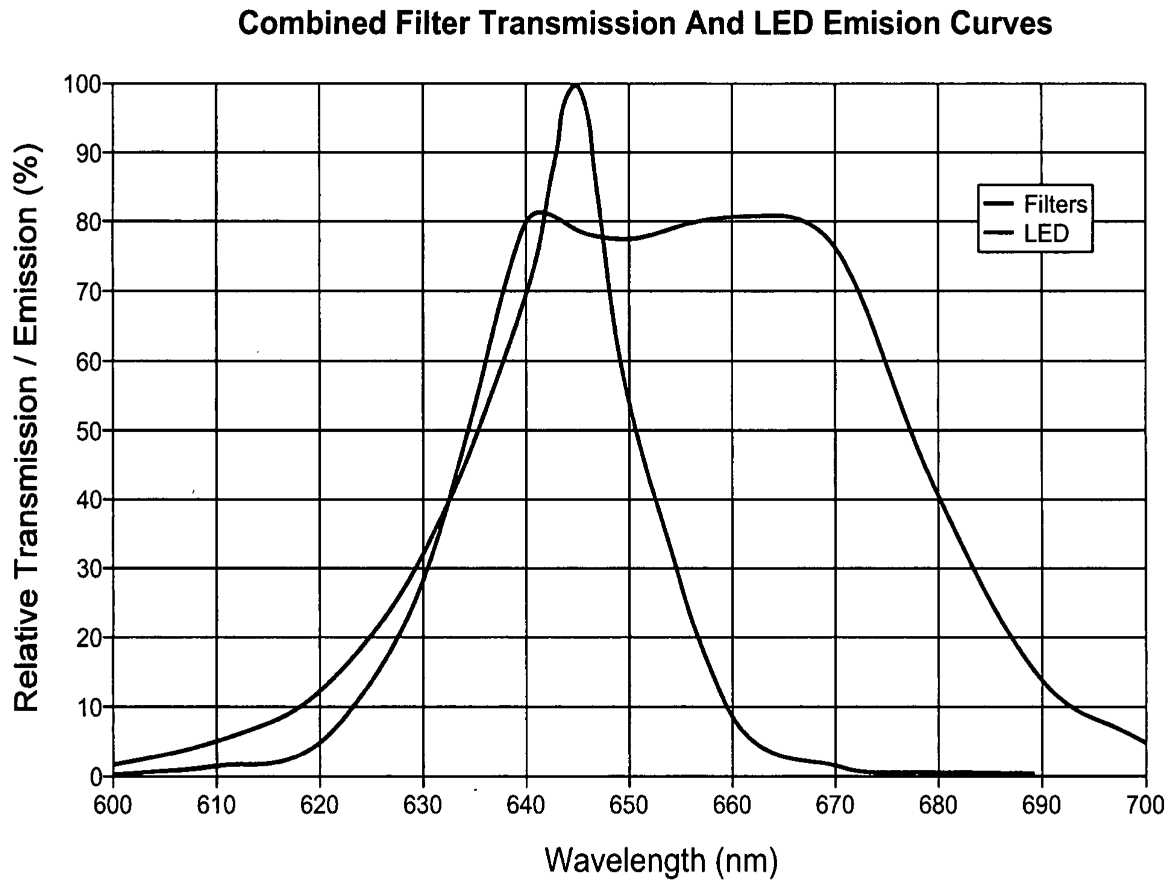
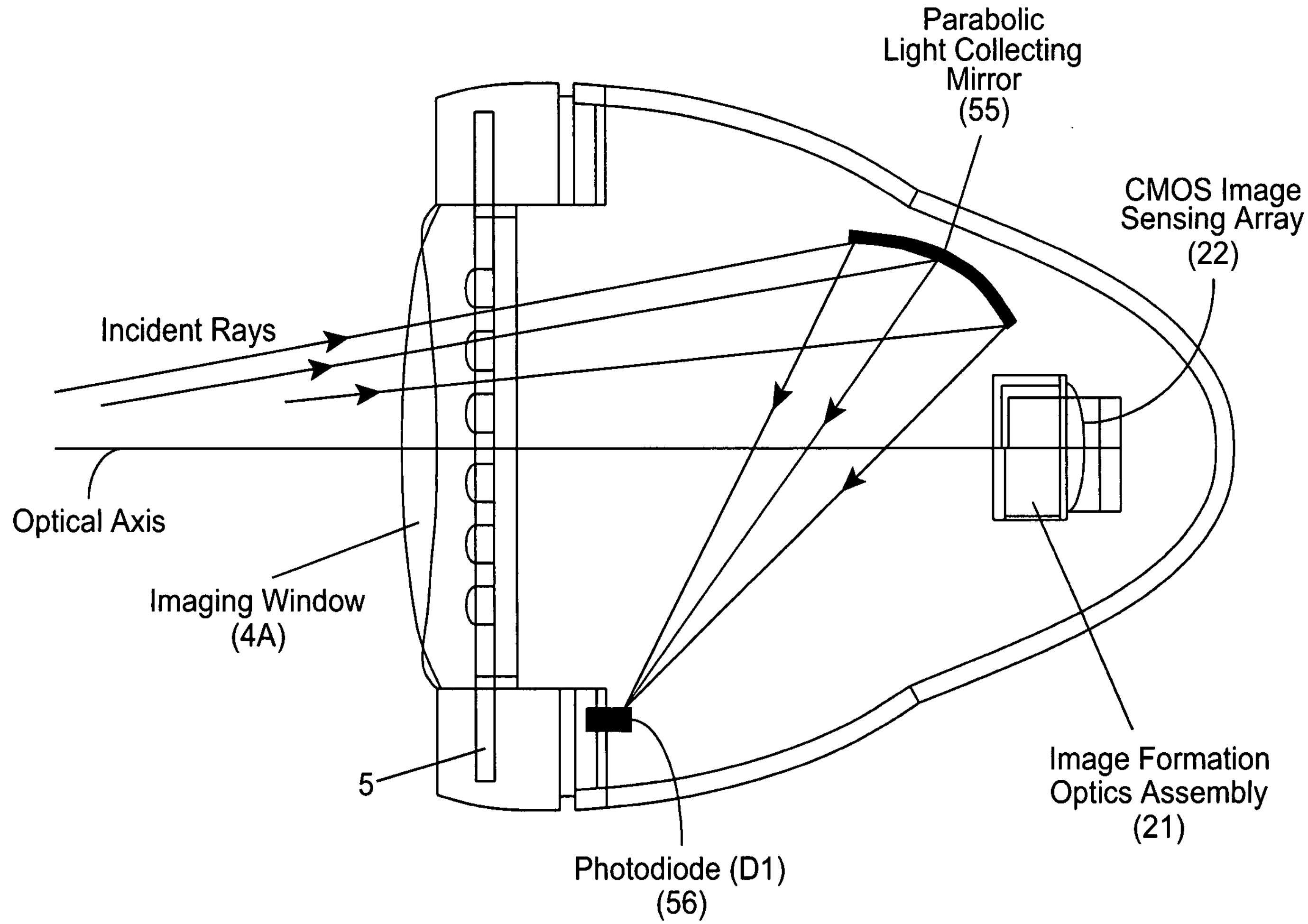


FIG. 6A3



($\Delta\lambda$) Bandwidth Of LED Emission Signal$\approx$15 nmeters

FIG. 6A4



Parabolic
Light Collecting
Mirror
(55)
CMOS Image
Sensing Array
(22)
Incident Rays
Optical Axis
Imaging Window
(4A)
5
Photodiode (D1)
(56)
Image Formation
Optics Assembly
(21)

FIG. 7A

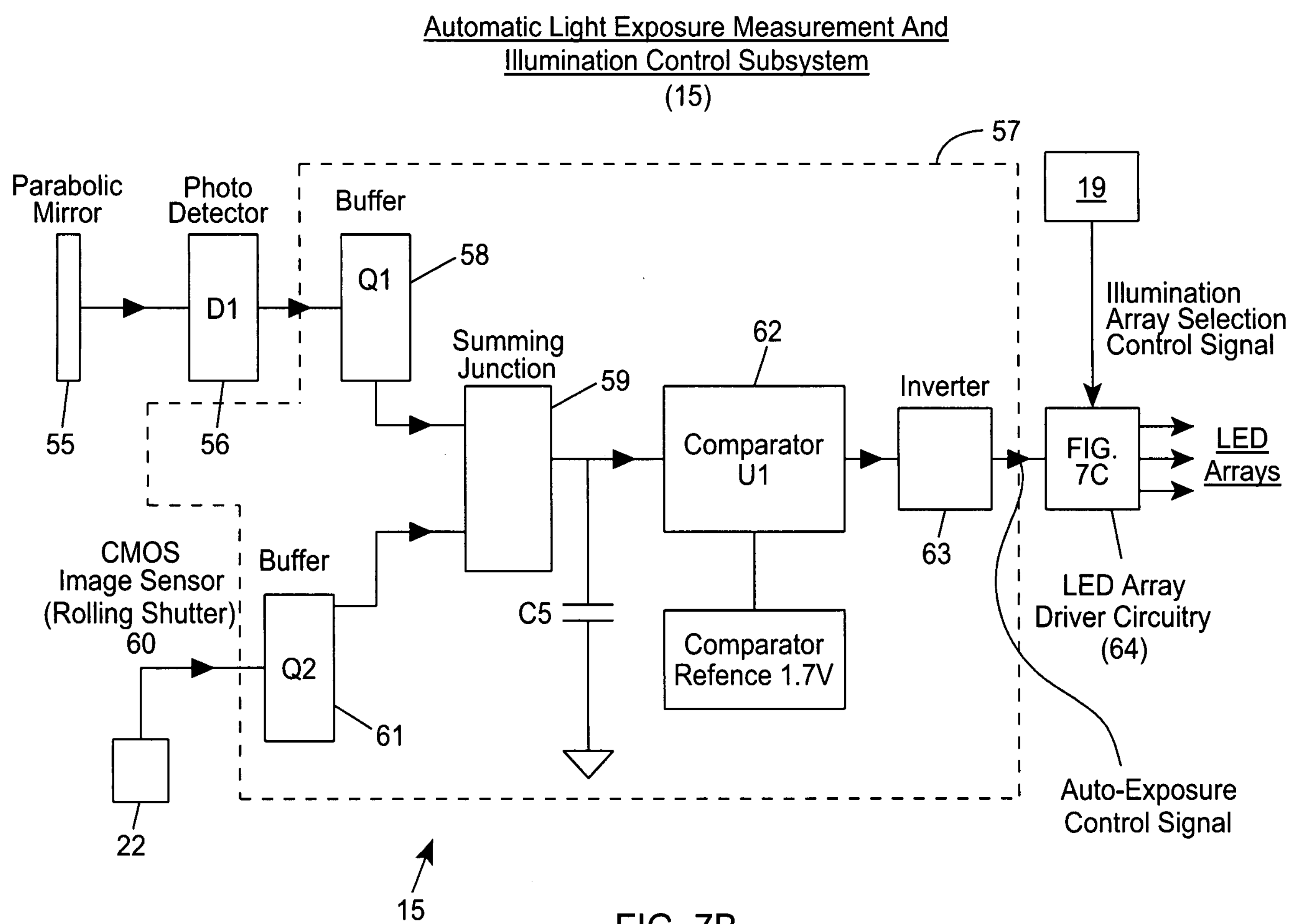

Automatic Light Exposure Measurement And
Illumination Control Subsystem
(15)
57
Parabolic Mirror
Photo Detector
D1
Buffer
Q1
58
55
56
Summing Junction
59
62
Comparator U1
Inverter
63
19
Illumination Array Selection Control Signal
FIG. 7C
LED Arrays
LED Array Driver Circuitry (64)
CMOS Image Sensor (Rolling Shutter) 60
Buffer
Q2
61
22
C5
Comparator Refence 1.7V
Auto-Exposure Control Signal
15

FIG. 7B

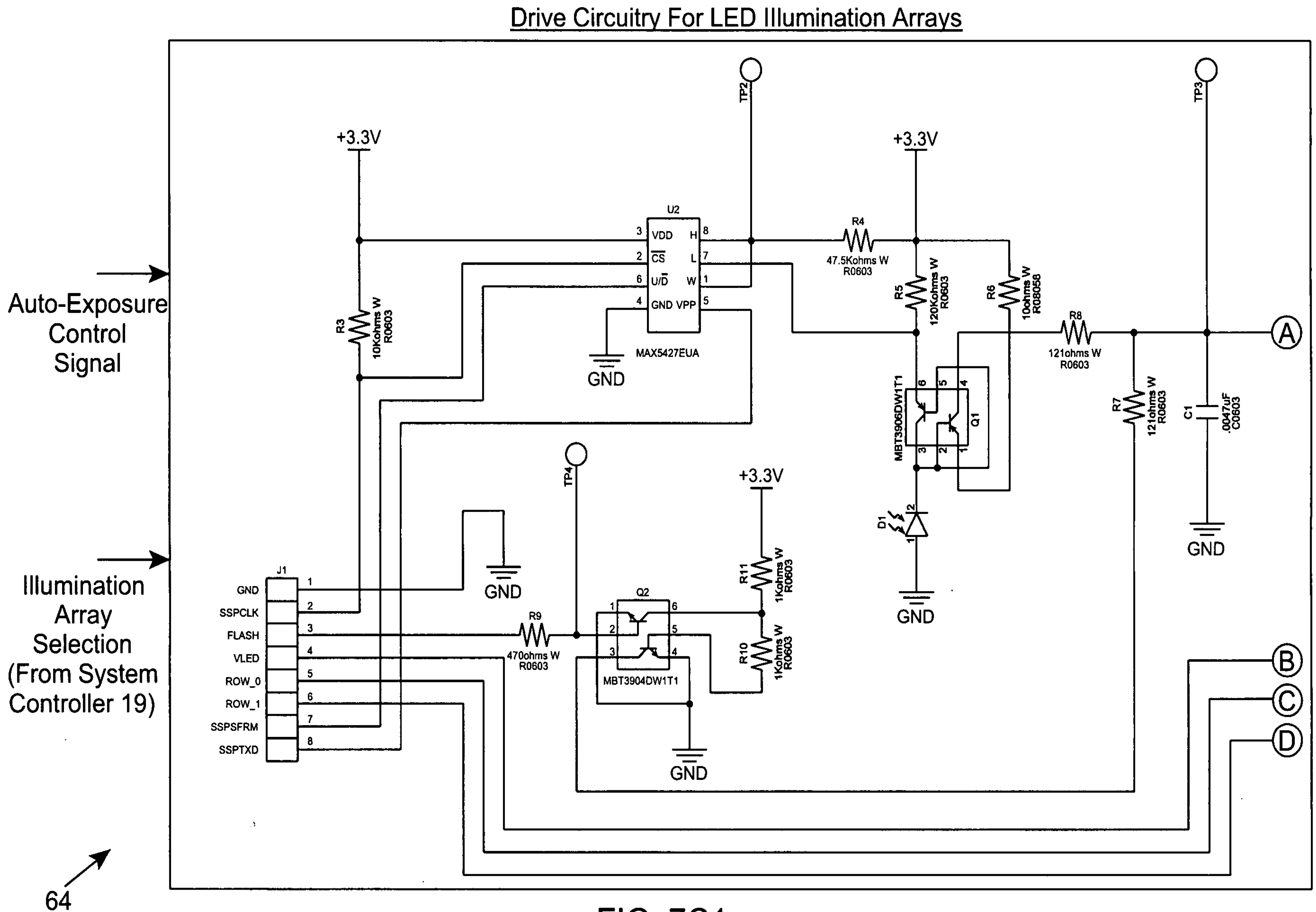
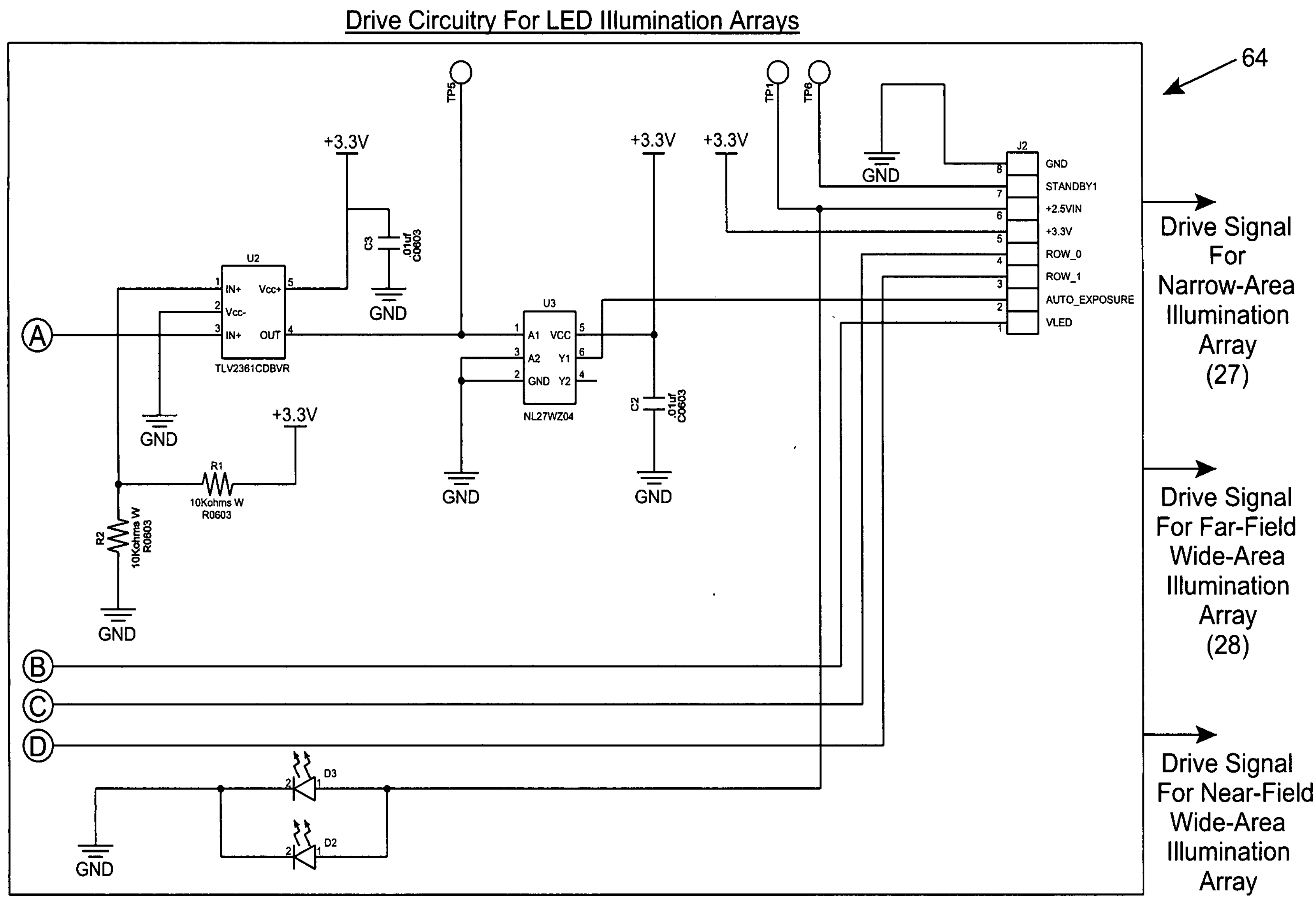

Drive Circuitry For LED Illumination Arrays
Auto-Exposure Control Signal
Illumination Array Selection (From System Controller 19)
64
+3.3V
U2
VDD
CS
U/D
GND VPP
H
L
W
MAX5427EUA
GND
TP2
TP3
TP4
R3
10Kohms W
R0603
R4
47.5Kohms W
R0603
R5
120Kohms W
R0603
R6
10ohms W
R08058
R8
121ohms W
R0603
R7
121ohms W
R0603
C1
.0047uF
C0603
MBT3906DW1T1
Q1
D1
A
B
C
D
J1
GND
SSPCLK
FLASH
VLED
ROW_0
ROW_1
SSPSFRM
SSPTXD
R9
470ohms W
R0603
Q2
MBT3904DW1T1
R11
1Kohms W
R0603
R10
1Kohms W
R0603

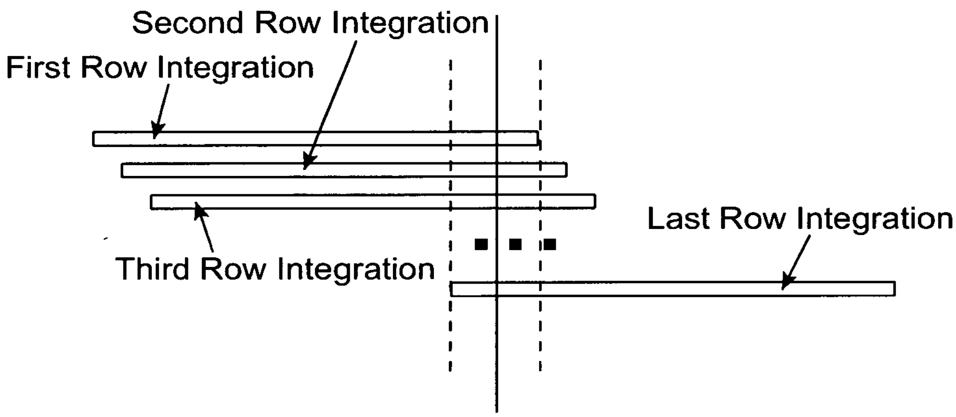
FIG. 7C1



Drive Circuitry For LED Illumination Arrays
64
TP5
TP1
TP6
+3.3V
+3.3V
+3.3V
+3.3V
GND
J2
GND
STANDBY1
+2.5VIN
+3.3V
ROW_0
ROW_1
AUTO_EXPOSURE
VLED
U2
IN+
Vcc-
IN+
Vcc+
OUT
TLV2361CDBVR
C3
.01uf
C0603
U3
A1
A2
GND
VCC
Y1
Y2
NL27WZ04
C2
.01uf
C0603
R1
10Kohms W
R0603
R2
10Kohms W
R0603
A
B
C
D
D3
D2
Drive Signal For Narrow-Area Illumination Array (27)
Drive Signal For Far-Field Wide-Area Illumination Array (28)
Drive Signal For Near-Field Wide-Area Illumination Array

FIG. 7C2

Global Exposure Control Method
Of Present Invention



b) Single Frame Shutter Mode

FIG. 7D

METHOD OF GLOBAL EXPOSURE CONTROL
WITHIN A IMAGING-BASED BAR CODE SYMBOL READING SYSTEM

A

STEP A: SELECT THE SINGLE FRAME SHUTTER MODE OF OPERATION FOR THE CMOS IMAGING SENSING ARRAY PROVIDED WITHIN AN IMAGING-BASED BAR CODE SYMBOL READING SYSTEM EMPLOYING AN AUTOMATIC LIGHT EXPOSURE MEASUREMENT AND ILLUMINATION CONTROL SUBSYSTEM, A MULTI-MODE ILLUMINATION SUBSYSTEM, AND A SYSTEM CONTROL SUBSYSTEM INTEGRATED THEREWITH, AND IMAGE FORMATION OPTICS PROVIDING THE CMOS IMAGE SENSING ARRAY WITH A FIELD OF VIEW INTO A REGION OF SPACE WHERE OBJECTS TO BE IMAGED ARE PRESENTED.

B

STEP B: USE THE AUTOMATIC LIGHT EXPOSURE MEASUREMENT AND ILLUMINATION CONTROL SUBSYSTEM TO CONTINUOUSLY COLLECT ILLUMINATION FROM A PORTION OF THE FIELD OF VIEW, DETECT THE INTENSITY OF THE COLLECTED ILLUMINATION, AND GENERATE AN ELECTRICAL ANALOG SIGNAL CORRESPONDING TO THE DETECTED INTENSITY, FOR PROCESSING.

C

STEP C: ACTIVATE (E.G. BY WAY OF THE SYSTEM CONTROL SUBSYSTEM 19 OR DIRECTLY BY WAY OF TRIGGER SWITCH 2C) THE CMOS IMAGE SENSING ARRAY SO THAT ITS ROWS OF PIXELS BEGIN TO INTEGRATE PHOTONICALLY GENERATED ELECTRICAL CHARGE IN RESPONSE TO THE FORMATION OF AN IMAGE ONTO THE CMOS IMAGE SENSING ARRAY BY THE IMAGE FORMATION OPTICS OF THE SYSTEM.

D

STEP D: WHEN ALL ROWS OF PIXELS IN THE IMAGE SENSING ARRAY ARE OPERATED IN A STATE OF INTEGRATION, AUTOMATICALLY GENERATE AN ELECTRONIC ROLLING SHUTTER (ERS) DIGITAL PULSE SIGNAL FROM THE CMOS IMAGE SENSING ARRAY AND PROVIDE THIS ERS PULSE SIGNAL TO THE AUTOMATIC LIGHT EXPOSURE MEASUREMENT AND ILLUMINATION CONTROL SUBSYSTEM SO AS TO ACTIVATE LIGHT EXPOSURE MEASUREMENT AND ILLUMINATION CONTROL OPERATIONS THEREWITHIN

A

FIG. 7E1

A

E: STEP E: UPON ACTIVATION OF THE AUTOMATIC LIGHT EXPOSURE MEASUREMENT AND ILLUMINATION CONTROL SUBSYSTEM, PROCESS THE ELECTRICAL ANALOG SIGNAL BEING CONTINUOUSLY GENERATED THEREWITHIN, MEASURE THE LIGHT EXPOSURE WITHIN A PORTION OF SAID FIELD OF VIEW, AND GENERATE AN AUTO-EXPOSURE CONTROL SIGNAL FOR CONTROLLING THE GENERATION OF ILLUMINATION FROM AT LEAST ONE LED-BASED ILLUMINATION ARRAY IN THE MULTI-MODE ILLUMINATION SUBSYSTEM THAT IS SELECTED BY AN ILLUMINATION ARRAY SELECTION CONTROL SIGNAL PRODUCED BY THE SYSTEM CONTROL SUBSYSTEM

↓

F: STEP: F: USE THE AUTO-EXPOSURE CONTROL SIGNAL AND THE ILLUMINATION ARRAY SELECTION CONTROL SIGNAL TO DRIVE THE SELECTED LED-BASED ILLUMINATION ARRAY AND GENERATE ILLUMINATION THEREFROM INTO THE FIELD OF VIEW OF THE CMOS IMAGE SENSING ARRAY, PRECISELY WHEN ALL ROWS OF PIXELS IN THE CMOS IMAGE SENSING ARRAY ARE IN A STATE OF INTEGRATION, THEREBY ENSURING THAT ALL ROWS OF PIXELS IN THE CMOS IMAGE SENSING ARRAY HAVE A COMMON INTEGRATION TIME.

FIG. 7E2

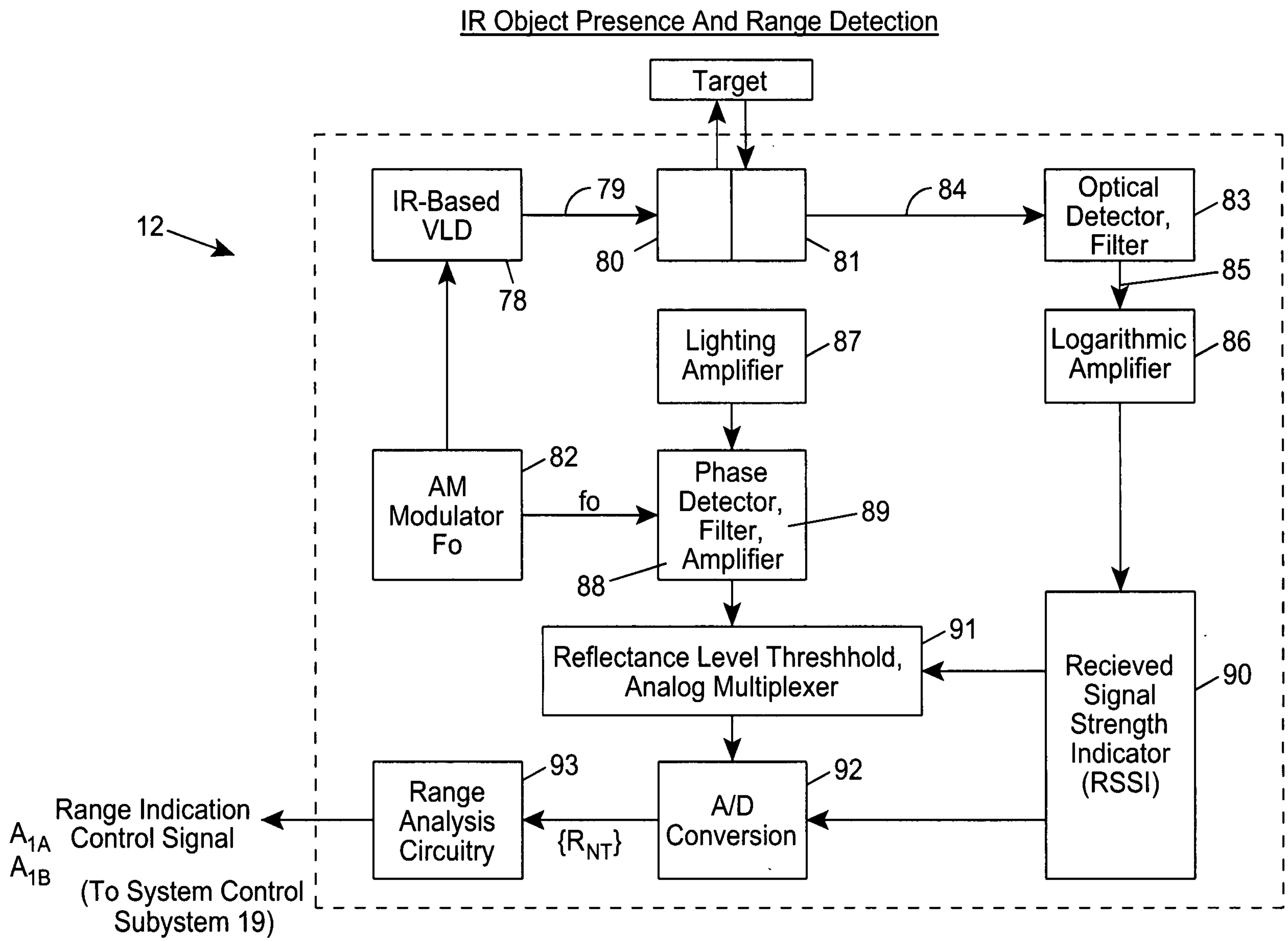

IR Object Presence And Range Detection
Target
IR-Based VLD
AM Modulator Fo
fo
Lighting Amplifier
Phase Detector, Filter, Amplifier
Optical Detector, Filter
Logarithmic Amplifier
Recieved Signal Strength Indicator (RSSI)
Reflectance Level Threshhold, Analog Multiplexer
A/D Conversion
{R_NT}
Range Analysis Circuitry
Range Indication Control Signal
A_1A
A_1B
(To System Control Subystem 19)
12
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93

FIG. 8

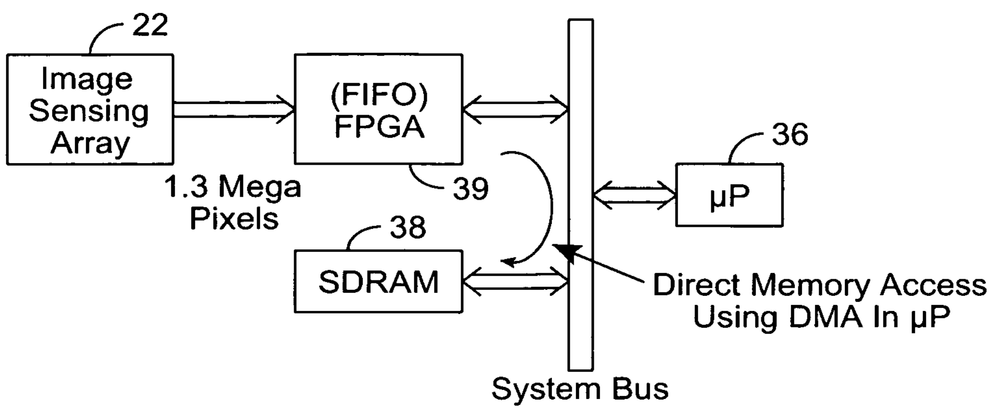

22
Image Sensing Array
(FIFO) FPGA
36
µP
1.3 Mega Pixels
39
38
SDRAM
Direct Memory Access Using DMA In µP
System Bus

FIG. 9

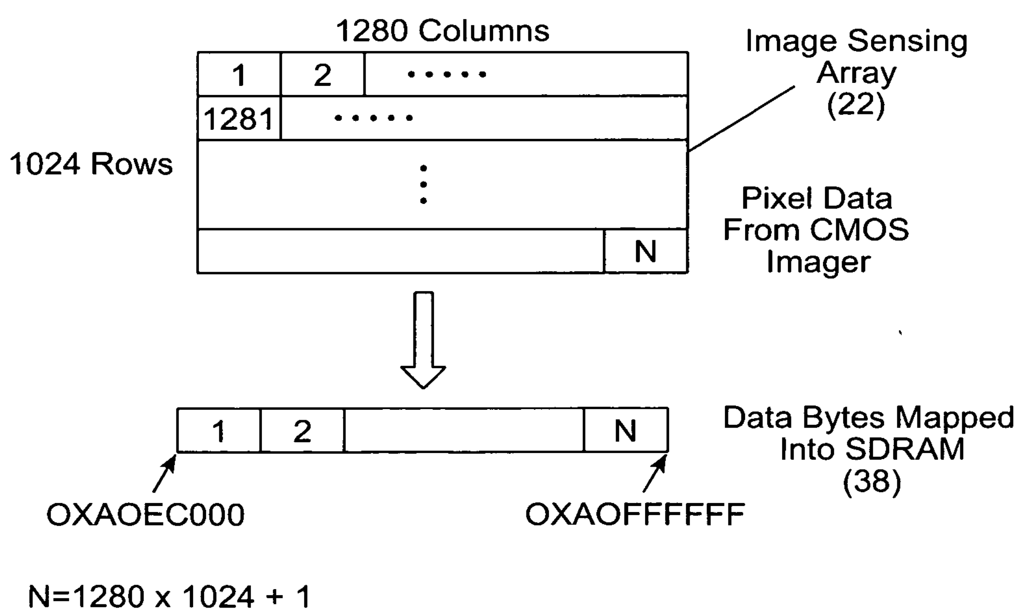
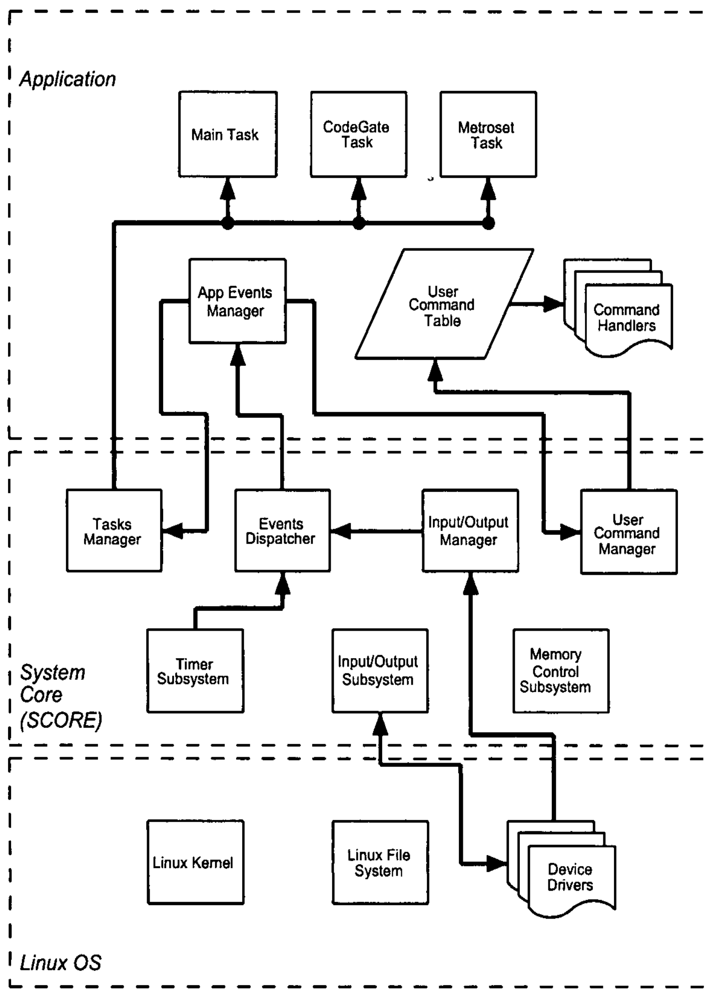

1280 Columns
1
2
• • • • •
1281
• • • • •
1024 Rows
Image Sensing Array (22)
Pixel Data From CMOS Imager
N
1
2
N
Data Bytes Mapped Into SDRAM (38)
OXAOEC000
OXAOFFFFFF
N=1280 x 1024 + 1

FIG. 10

Software Block Diagram



3-Tier Software Architecture:

- Linux OS
- System Core (SCORE) Software
- Product Application Software

FIG. 11

Events Dispatcher

Provides a means of signaling and delivering events to the App Events Manager

(pointer to App Events Manager is provided at the SCORE initialization)

```
int
ScoreSignalEvent(int event_id,          /* Input: event id */
          void * p_par);          /* Input: pointer to the event's parameters */
```

App Events Manager is responsible for processing the event: It can start a new task, or stop currently running task, or do something or nothing and simply ignore the event.

FIG. 12A

Examples of System-Defined Events

SCORE_EVENT_POWER_UP

Signals the completion of the system start-up. No parameters.

SCORE_EVENT_TIMEOUT

Signals the timeout of the logical timer. Parameter: pointer to timer id.

SCORE_EVENT_UNEXPECTED_INPUT

Signals that the unexpected input data is available. Parameter: pointer to connection id.

SCORE_EVENT_TRIG_ON

Signals that the user pulled the trigger. No parameters.

SCORE_EVENT_TRIG_OFF

Signals that the user released the trigger. No parameters.

SCORE_EVENT_OBJECT_DETECT_ON

Signals that the object is positioned under the camera. No parameters.

SCORE_EVENT_OBJECT_DETECT_OFF

Signals that the object is removed from the field-of view of the camera. No parameters.

SCORE_EVENT_EXIT_TASK and **SCORE_EVENT_ABORT_TASK**

Signal the end of the task execution. Parameter: pointer to the UTID.

FIG. 12B

Tasks Manager

Provides a means of executing and stopping application specific tasks (threads)

```
typedef void *                              /* Return: pointer to the set of returned parameters */
(*TASK_FUNC)(void *params);                 /* Input:  set of input parameters */

int                                         /* Return: 0 if successful, otherwise error code */
ScoreStartTask(TASK_FUNC task_func,/* Input:  pointer to the task's main function */
      int task_id,                          /* Input:  id assigned to the task by application */
     void *task_params,                     /* Input:  parameters passed to the task's main function */
     int task_owner,                        /* Input:  connection id of the task's owner */
     int task_priority,                     /* Input:  task's priority (must be 0 for now) */
     size_t stacksize,                      /* Input:  size of the stack, or 0 for default size */
     size_t heapsize,                       /* Input:  size of the heap, or 0 for default size */
     UTID *p_utid);                         /* Output: unique task identifier */

BOOL      /* Return: TRUE if it kills the task, or FALSE if the task was not found */
ScoreKillTask(UTID pthread_id)              /* Input:  unique task identifer */
```

FIG. 12C

Input / Output Manager

- High priority thread running in the background and monitoring activities of the external devices and user connections
- Signals appropriate events to the application when such activities are detected

FIG. 12D

Input / Output Subsystem

Provides a means of creating and deleting input/output connections...

```
int                                     /* Return: connection id if successful, otherwise (-1) */
ScoreIomngrCreateConnection(int conn_type,      /* Input:  connection type */
    int fd,                             /* Input:  file descriptor of a device or a socket */
    int conn_state,                     /* Input:  initial state of the connection, the value controlled by application */
    void *properties);                          /* Input:  pointer to the connection properties */

int                                     /* Return: connection id if successful, otherwise (-1) */
ScoreInitRS232(char *full_name,                 /* Input:  full name of the device, such as "/dev/ttyS0" */
        RS232_PROP *rs232_prop);                /* Input: RS232 parameters */
```

FIG. 12E1

Input / Output Subsystem

...and communicating with the outside world

```
int                                          /* Return: number of bytes received */
ScoreIomngrGetData(int connection_id,  /* Input:  connection id, or -1 for the task owner */
            char *input_buffer,              /* Input:  pointer to the input buffer */
            int min_len,             /* Input:  minimum number of bytes to receive */
            int max_len,             /* Input:  maximum number of bytes to receive */
            BOOL echo,               /* Input:  TRUE if data should be echoed back to device, otherwise FALSE */
            int timeout_ms);         /* Input:  If not 0, number of milliseconds to wait */

int                                          /* Return: 0 if successful, or (-1) in case of error */
ScoreIomngrSendData(int connection_id,               /* Input:  connection id */
            char *p_data,            /* Input:  pointer to the data buffer */
            int len);                /* Input:  number of bytes to send */

void
ScoreIomngrSendStream(int stream_type,               /* Input:  type of output stream */
            char *p_data,                            /* Input:  pointer to the data buffer */
            int len);                                /* Input:  number of bytes to send */
```

FIG. 12E2

Timer Subsystem

Provides a means of creating, deleting...

```
int                                  /* Return: timer id if successful, otherwise (-1) */
ScoreCreateTimer(int flags);         /* Input:  optional SCORE_TIMER_CONTINUOUS */

void
ScoreDeleteTimer(int timer_id);             /* Input:  timer id, must be >= 0 */

int                                  /* Return: 0 if successful, otherwise (-1) */
ScoreStartTimer(int timer_id,        /* Input:  timer id */
        int time_ms);                /* Input:  timer value, in ms */

int                                  /* Return: 0 if successful, otherwise (-1) */
ScoreStopTimer(int timer_id);        /* Input:  timer id */
```

FIG. 12F1

Timer Subsystem

...and utilizing logical timers

```
BOOL                          /* Return: TRUE if the timer timed out, otherwise FALSE */
ScoreTimerTimedOut(int timer_id);       /* Input:  timer id */

int                    /* Return: time (in ms) left before the timer times out, or (-1) in case of error */
ScoreGetTimeLeft(int timer_id);                     /* Input:  timer id */

int          /* Return: time (in ms) gone since the timer has been started (or restarted), or (-1) in case of error */
ScoreGetTime(int timer_id);                 /* Input:  timer id */

BOOL                   /* Return: TRUE if timer is stopped, otherwise FALSE */
ScoreIsTimerStopped(int timer_id);      /* Input:  timer id */
```

FIG. 12F2

Memory Control Subsystem

Provides a thread-level dynamic memory management (the interfaces fully compatible with standard dynamic memory management functions)...

```
void *                        /* Return: pointer to the allocated memory if successful, otherwise NULL */
ScoreMalloc(size_t size);     /* Input:  size, in bytes, of the needed memory */

void
ScoreFree(void *mem);         /* Input:  pointer to the memory to be freed */
```

FIG. 12G1

Memory Control Subsystem

...as well as a means of buffering the data

```
int                                                          /* Return: 0 if successful */
ScoreCreateOutpMem(SCORE_OUTP_MEM *p_outp_mem);              /* Input: pointer to buffered memory structure */

void
ScoreDestroyOutpMem(SCORE_OUTP_MEM  *p_outp_mem);            /* Input: pointer to buffered memory structure */

int                                                  /* Return: 0 if successful */
ScoreWriteToOutpMem (SCORE_OUTP_MEM *p_outp_mem,             /* Input: pointer to buffered memory structure */
     void *p_data,                                   /* Input: pointer to the data to be buffered up for output */
     size_t len);                                    /* Input: size of the data, in bytes */

int                                                  /* Return: 0 if successful */
ScoreSendDataFromOutpMem(int connection_id,          /* Input: id of the connection to send the data to */
      SCORE_OUTP_MEM  *p_outp_mem);                  /* Input: pointer to buffered memory structure */

int                                                  /* Return: 0 if successful */
ScoreSendStreamFromOutpMem(int stream_type,          /* Input:  type of output stream */
      SCORE_OUTP_MEM  *p_outp_mem);                  /* Input: pointer to buffered memory structure */
```

FIG. 12G2

User Commands Manager

Provides a standard way of entering user commands and executing application modules responsible for handling them

(pointer to User Commands Table is provided at the SCORE initialization)

```
int
ScoreCmdManager(void *params);

rc = ScoreStartTask(ScoreCmdManager,      /* Input:  user command manager task */
       CMDMNGR_TASK_ID,                   /* Input:  id assigned to the commands manager */
       NULL,
       0,
       connection_id,                     /* Input:  connection id of the owner */
       0,                                 /* Input:  priority */
       (64 * 1024),                       /* Input:  stack size */
       (512 * 1024),                      /* Input:  heap size */
       &cmdmngr_utid);                    /* Output: unique task identifier */
```

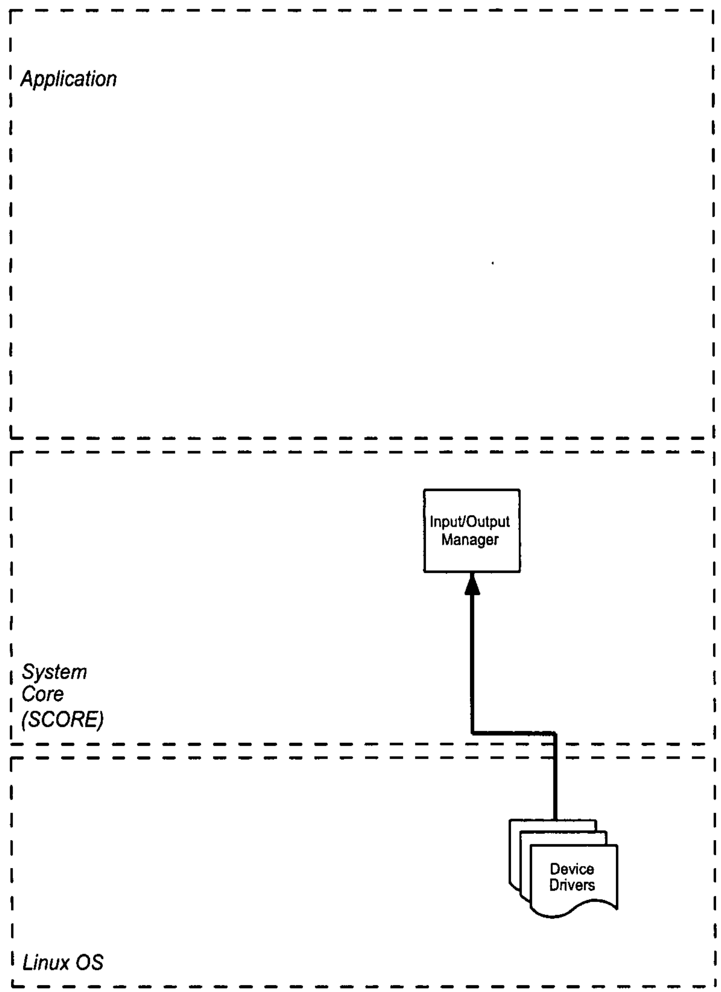
FIG. 12H

Device Drivers

- Trigger driver -- establishes software connection with the hardware trigger
- Image acquisition driver -- implements image acquisition functionality
- IR driver -- implements object detection functionality

FIG. 12I

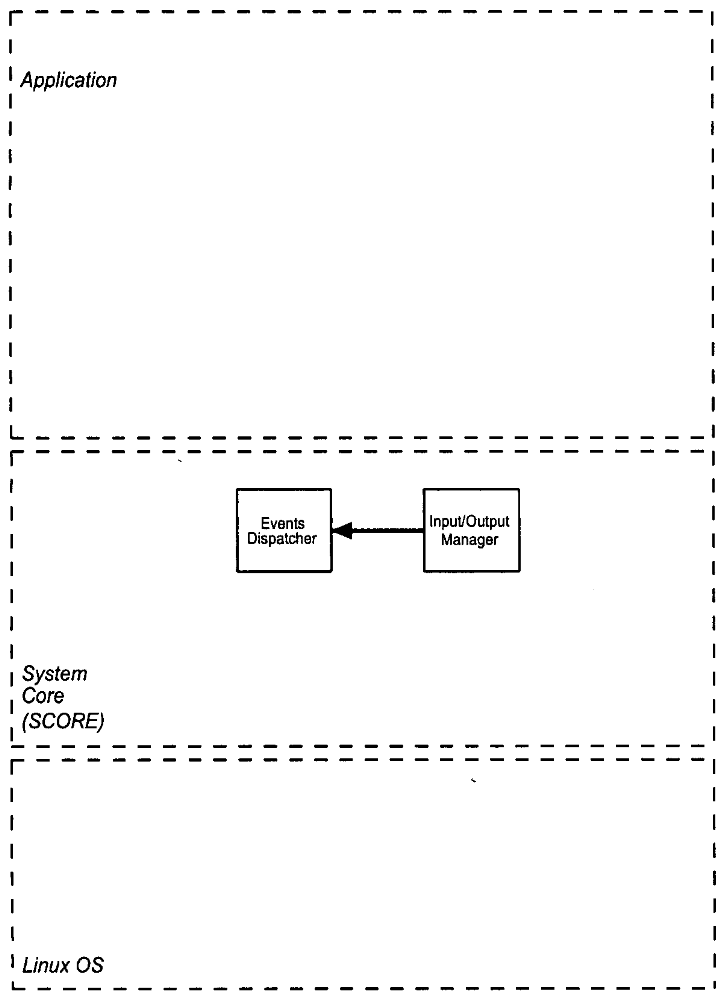
Example of Flow of Events



- User points the scanner towards a barcode label
- Object is detected
- The IR device driver wakes up the Input/Output Manager

FIG. 13A

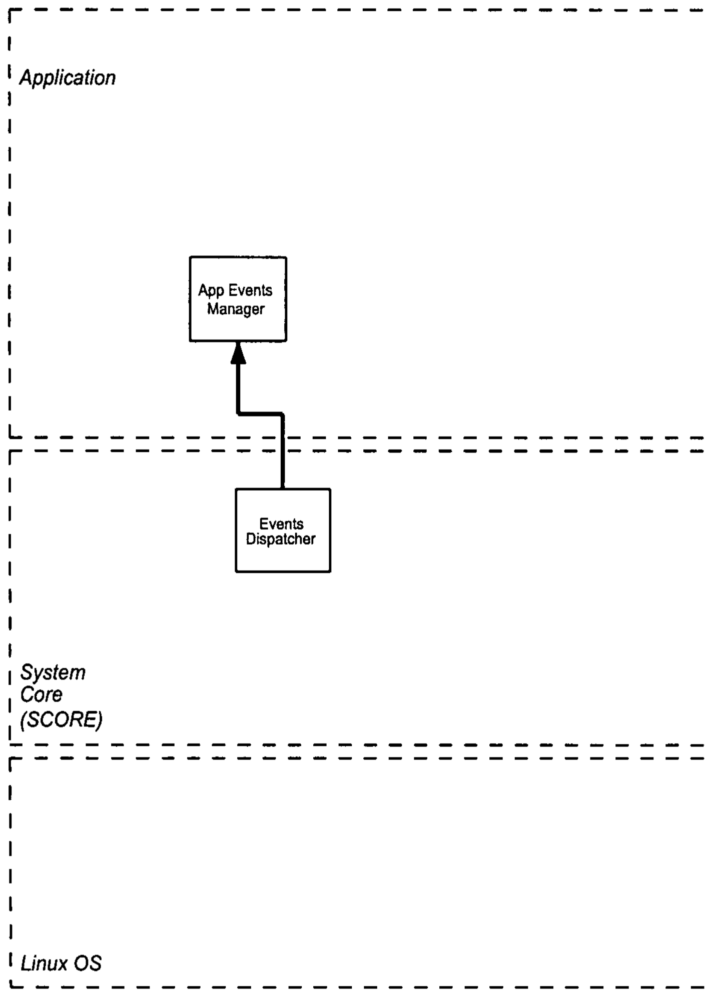
Example of Flow of Events



- The Input/Output Manager posts the SCORE_OBJECT_DETECT_ON event

FIG. 13B

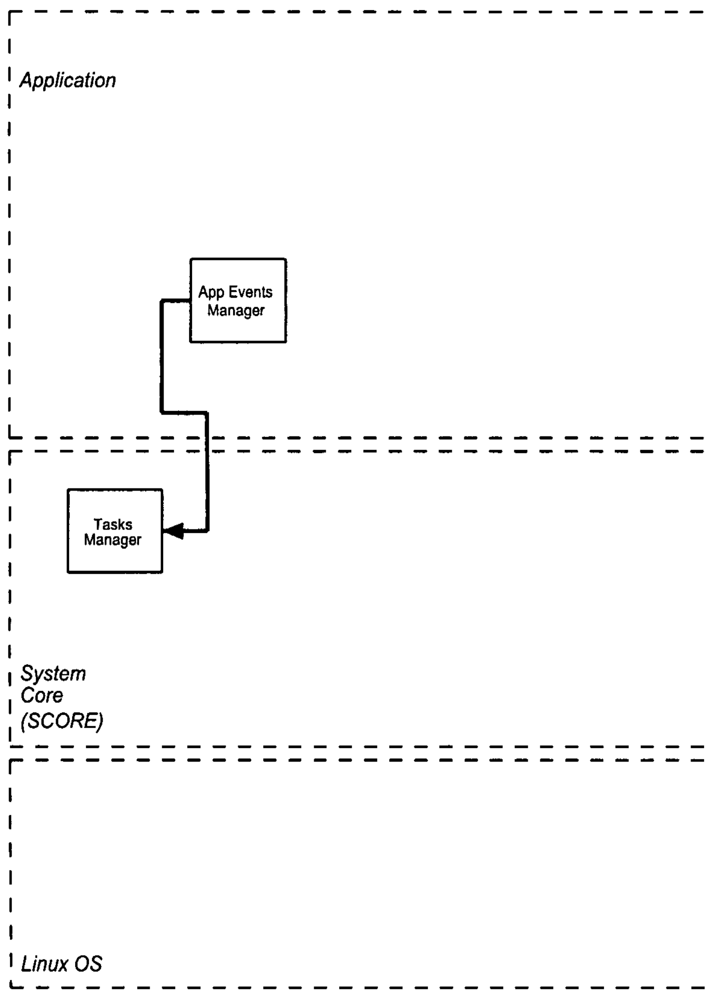
Example of Flow of Events

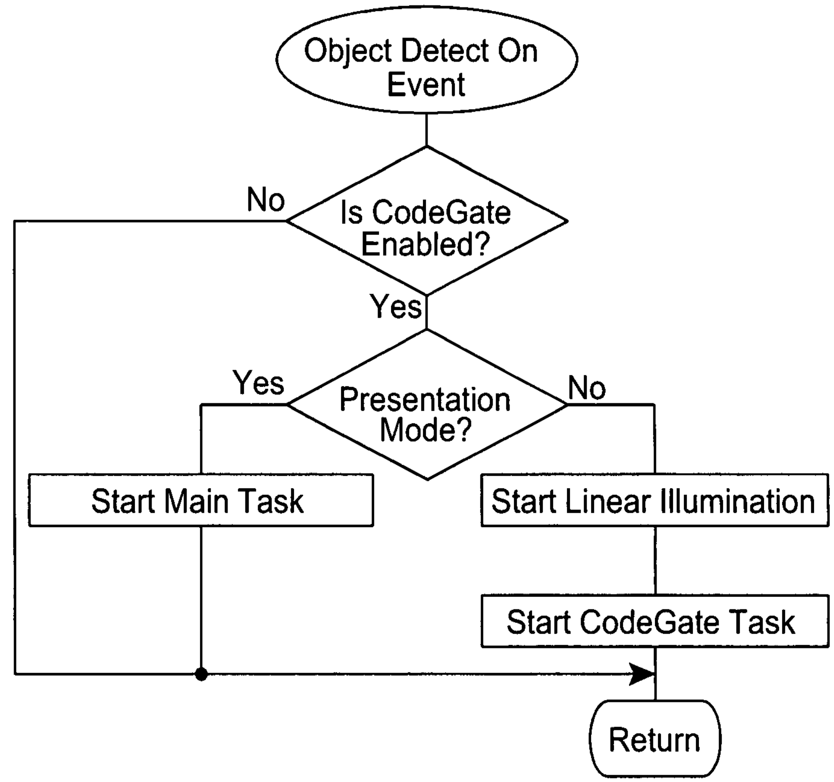


- The Events Dispatcher passes the SCORE_OBJECT_DETECT_ON event to the application

FIG. 13C

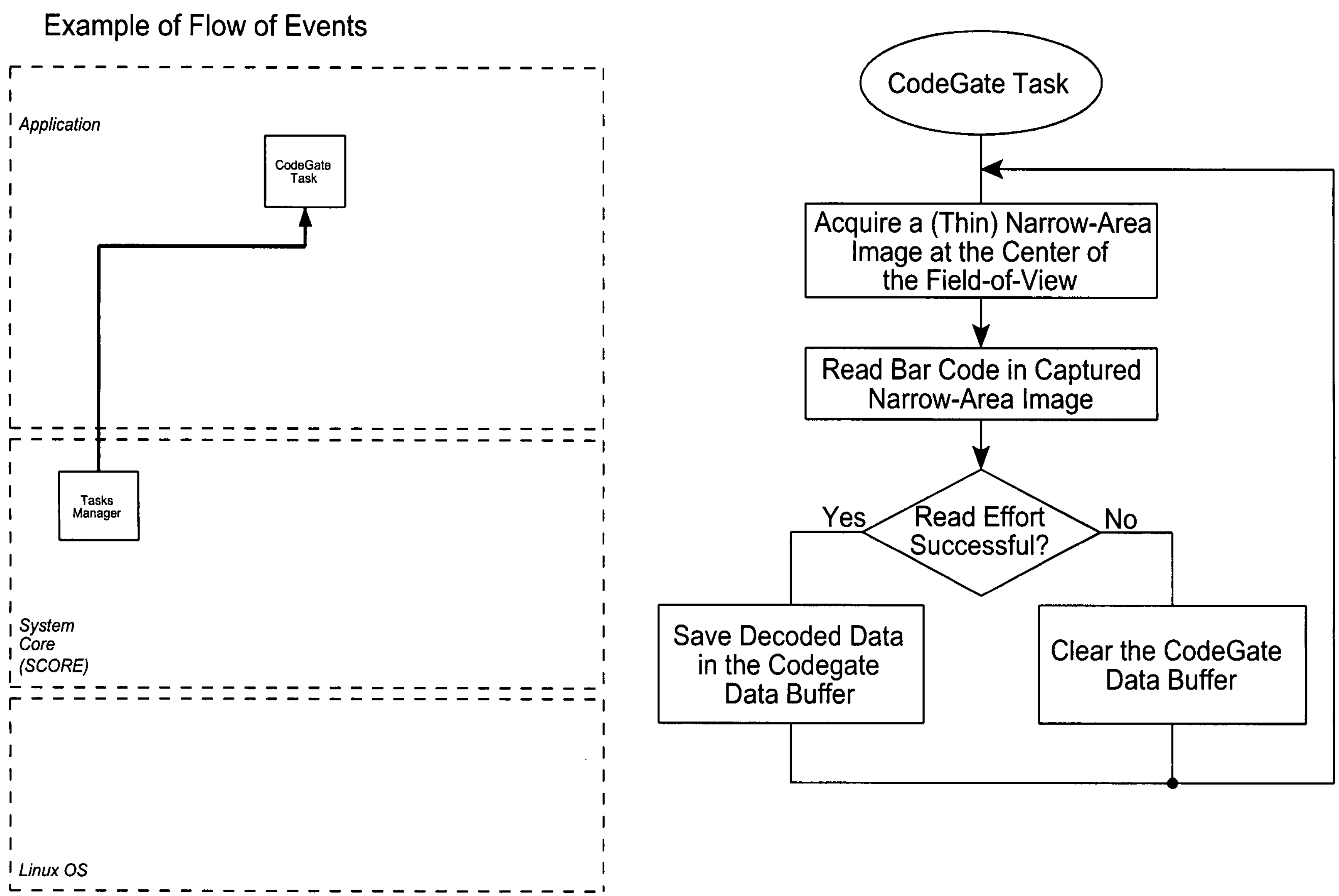
Example of Flow of Events





- The SCORE_OBJECT_DETECT_ON event handling routine starts linear illumination and executes the CodeGate Task

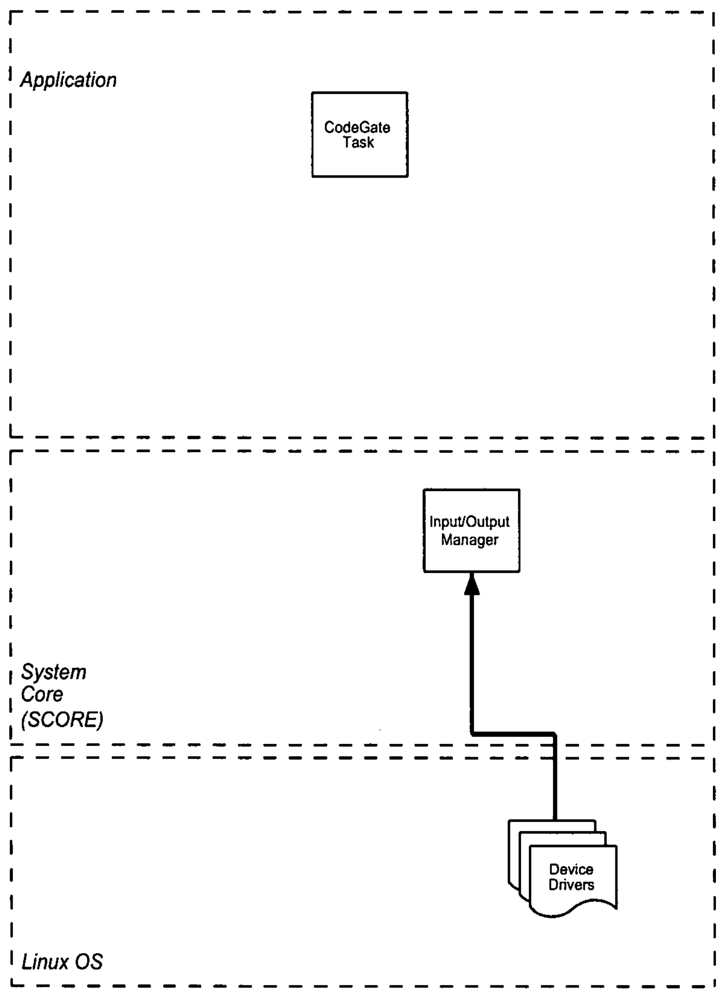
FIG. 13D



Example of Flow of Events
Application
CodeGate Task
Tasks Manager
System Core (SCORE)
Linux OS
CodeGate Task
Acquire a (Thin) Narrow-Area Image at the Center of the Field-of-View
Read Bar Code in Captured Narrow-Area Image
Read Effort Successful?
Yes
No
Save Decoded Data in the Codegate Data Buffer
Clear the CodeGate Data Buffer

FIG. 13E

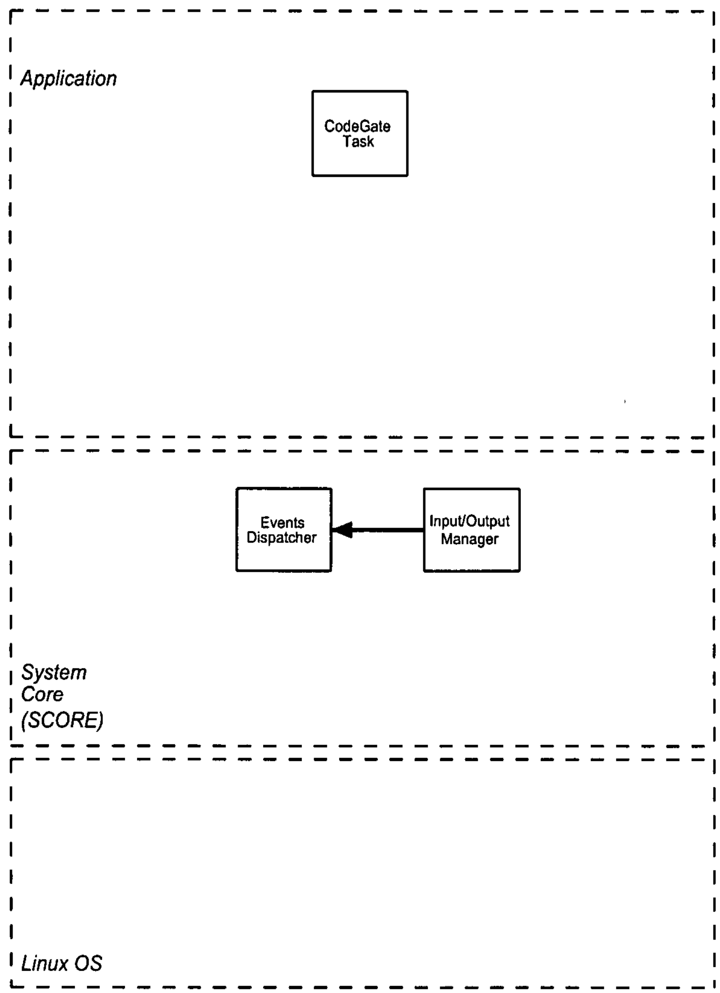
Example of Flow of Events



- User pulls the trigger
- The trigger device driver wakes up the Input/Output Manager

FIG. 13F

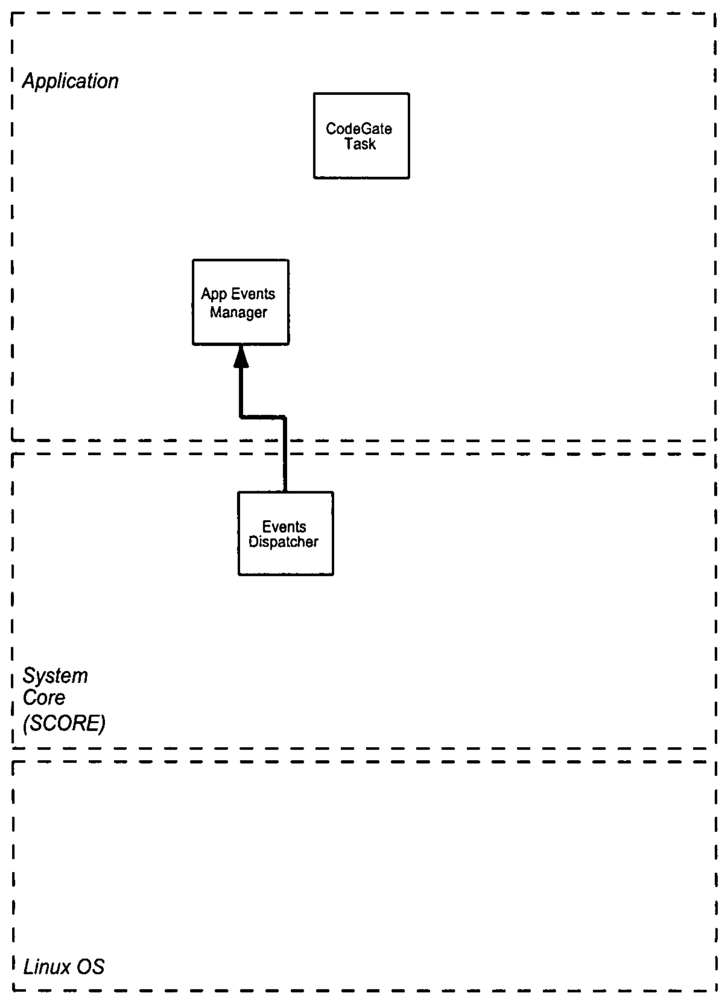
Example of Flow of Events



- The Input/Output Manager posts the SCORE_TRIGGER_ON event

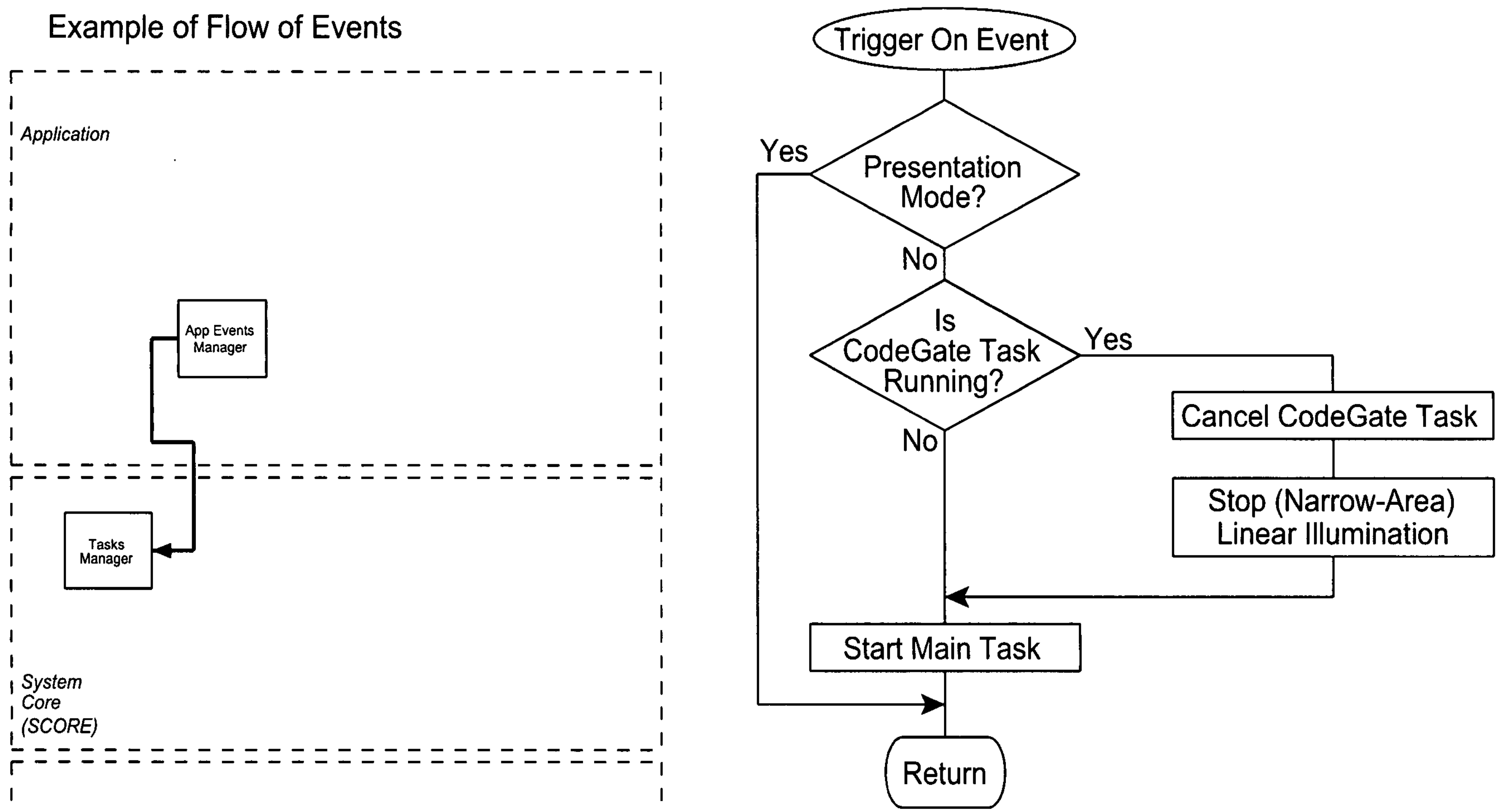
FIG. 13G

Example of Flow of Events



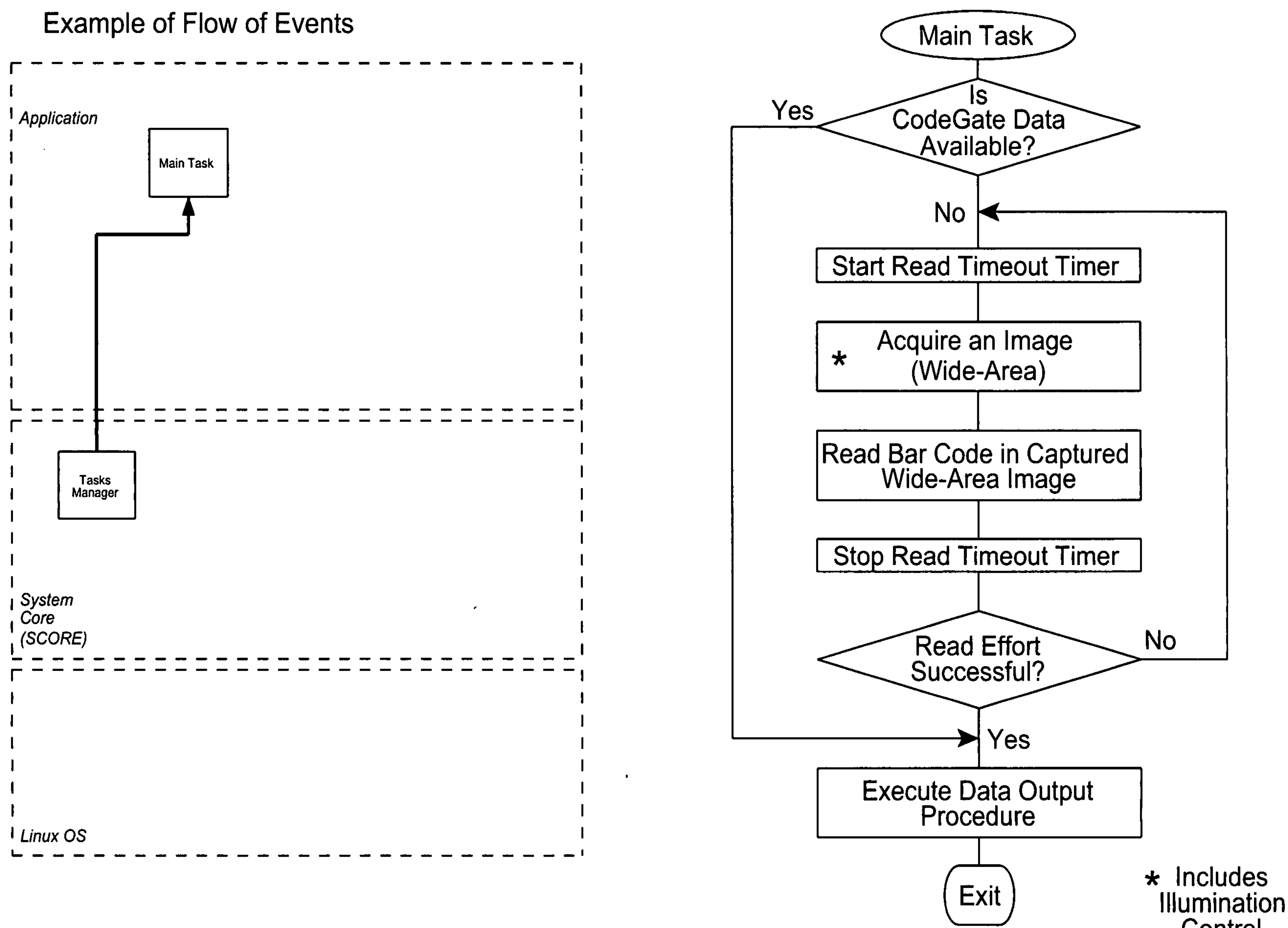
- The Events Dispatcher passes the SCORE_TRIGGER_ON event to the application

FIG. 13H



- The SCORE_TRIGGER_ON event handling routine stops linear illumination, cancels the CodeGate Task, and executes the Main Task

FIG. 13I



Example of Flow of Events
Application
Main Task
Tasks Manager
System Core (SCORE)
Linux OS
Main Task
Is CodeGate Data Available?
Yes
No
Start Read Timeout Timer
* Acquire an Image (Wide-Area)
Read Bar Code in Captured Wide-Area Image
Stop Read Timeout Timer
Read Effort Successful?
No
Yes
Execute Data Output Procedure
Exit
* Includes Illumination Control

FIG. 13J

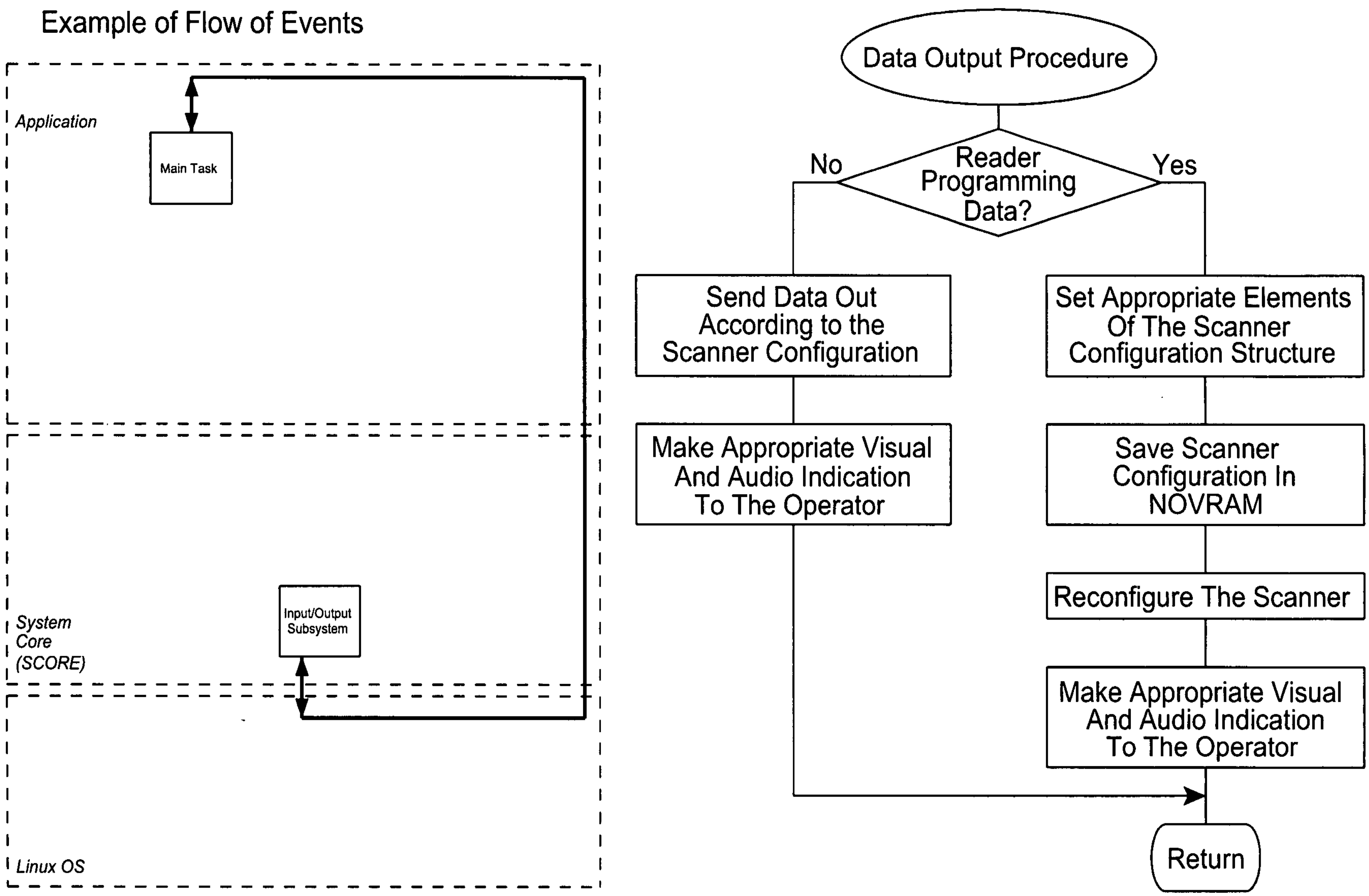
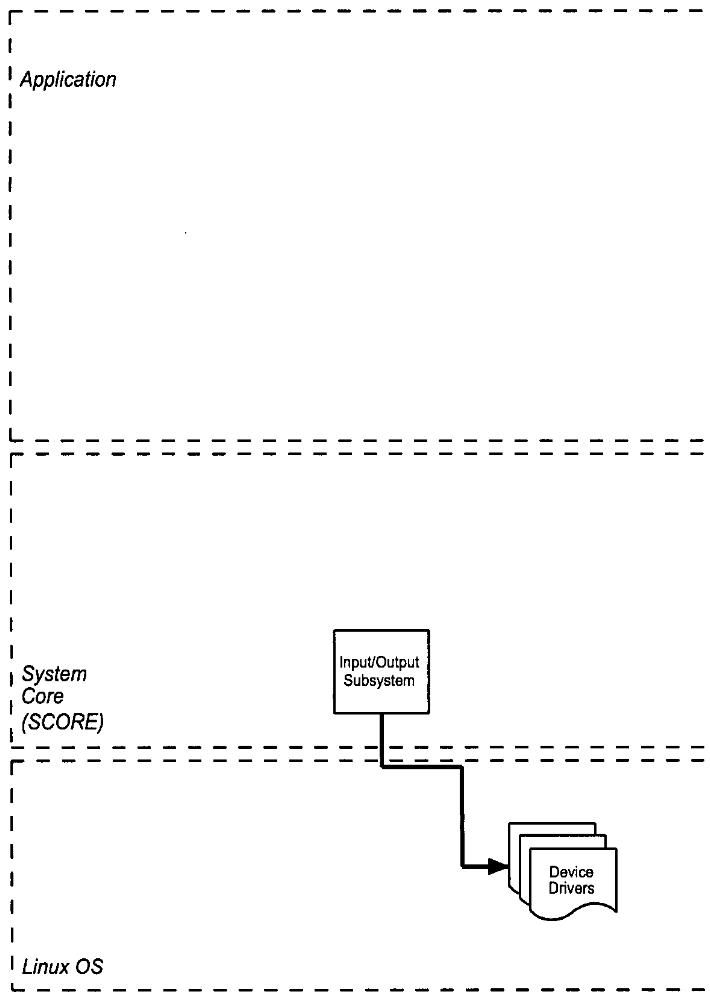

Example of Flow of Events
Application
Main Task
System Core (SCORE)
Input/Output Subsystem
Linux OS
Data Output Procedure
Reader Programming Data?
No
Yes
Send Data Out According to the Scanner Configuration
Make Appropriate Visual And Audio Indication To The Operator
Set Appropriate Elements Of The Scanner Configuration Structure
Save Scanner Configuration In NOVRAM
Reconfigure The Scanner
Make Appropriate Visual And Audio Indication To The Operator
Return

FIG. 13K

Example of Flow of Events



• The decoded data is sent to the user

FIG. 13L

METHOD OF ILLUMINATING OBJECTS WITHOUT SPECULAR REFLECTION

STEP A: USE THE AUTOMATIC LIGHT EXPOSURE MEASUREMENT AND CONTROL SUBSYSTEM TO MEASURE THE LIGHT LEVEL TO WHICH THE CMOS IMAGE SENSING ARRAY IS EXPOSED.

STEP B: USE THE AUTOMATIC IR-BASED OBJECT PRESENCE AND RANGE DETECTION SUBSYSTEM TO MEASURE THE PRESENCE AND RANGE OF THE OBJECT IN EITHER THE NEAR OR FAR FIELD PORTION OF THE FIELD OF VIEW (FOV) OF THE SYSTEM.

STEP C: USE THE DETECTED RANGE AND THE MEASURED LIGHT EXPOSURE LEVEL TO DRIVE BOTH THE UPPER AND LOWER LED SUBARRAYS ASSOCIATED WITH EITHER THE NEAR OR FAR FIELD WIDE AREA ILLUMINATION ARRAY.

STEP D: CAPTURE A WIDE-AREA IMAGE AT THE CMOS IMAGE SENSING ARRAY USING THE ILLUMINATION FIELD PRODUCED DURING STEP C.

STEP E: RAPIDLY PROCESS THE CAPTURED WIDE-AREA IMAGE DURING STEP D TO DETECT THE OCCURANCE OF HIGH SPATIAL-INTENSITY LEVELS IN THE CAPTURED WIDE-AREA IMAGE, INDICATIVE OF A SPECULAR REFLECTION CONDITION.

STEP F:

IF A SPECULAR REFLECTION CONDITION IS DETECTED IN THE PROCESSED WIDE-AREA IMAGE, THEN DRIVE ONLY THE UPPER LED SUBARRAY ASSOCIATED WITH EITHER THE NEAR FIELD OR FAR FIELD WIDE AREA ILLUMINATION ARRAY.

IF A SPECULAR REFLECTION CONDITION IS NOT DETECTED IN THE PROCESSED WIDE-AREA IMAGE, THEN USE THE DETECTED RANGE AND THE MEASURED LIGHT EXPOSURE LEVEL TO DRIVE BOTH THE UPPER AND LOWER LED SUBARRAYS ASSOCIATED WITH EITHER THE NEAR FIELD OR FAR FIELD WIDE AREA ILLUMINATION ARRAY.

FIG. 13M1

STEP G: CAPTURE A WIDE-AREA IMAGE AT THE CMOS IMAGE SENSING ARRAY USING THE ILLUMINATION FIELD PRODUCED DURING STEP F.

STEP H: RAPIDLY PROCESS THE CAPTURED WIDE-AREA IMAGE DURING STEP G TO DETECT THE OCCURANCE OF HIGH SPATIAL-INTENSITY LEVELS IN THE CAPTURED WIDE-AREA IMAGE, INDICATIVE OF A SPECULAR REFLECTION CONDITION.

STEP I:

IF A SPECULAR REFLECTION CONDITION IS STILL DETECTED IN THE PROCESSED WIDE-AREA IMAGE, THEN DRIVE THE OTHER LED SUBARRAY ASSOCIATED WITH EITHER THE NEAR FIELD OR FAR FIELD WIDE AREA ILLUMINATION ARRAY.

IF A SPECULAR REFLECTION CONDITION IS NOT DETECTED IN THE PROCESSED WIDE-AREA IMAGE, THEN DRIVE USE THE DETECTED RANGE AND THE MEASURED LIGHT EXPOSURE LEVEL TO DRIVE THE SAME LED SUBARRAY (AS IN STEP C) ASSOCIATED WITH EITHER THE NEAR FIELD OR FAR FIELD WIDE AREA ILLUMINATION ARRAY.

STEP J: CAPTURE A WIDE-AREA IMAGE AT THE CMOS IMAGE SENSING ARRAY USING THE ILLUMINATION FIELD PRODUCED DURING STEP I.

STEP K: RAPIDLY PROCESS THE CAPTURED WIDE-AREA IMAGE DURING STEP J TO DETECT THE ABSENCE OF HIGH SPATIAL-INTENSITY LEVELS IN THE CAPTURED WIDE-AREA IMAGE, CONFIRMING THE ELIMINATION OF THE ONCE DETECTED SPECULAR REFLECTION CONDITION.

FIG. 13M2

STEP L:

IF NO SPECULAR REFLECTION CONDITION IS DETECTED IN THE PROCESSED WIDE-AREA IMAGE AT STEP K, THEN PROCESS THE WIDEAREA IMAGE USING MODE(S) SELECTED FOR THE MULTI-MODE IMAGEPROCESSING BAR CODE READING SUBSYSTEM.

IF A SPECULAR REFLECTION CONDITION IS STILL DETECTED IN THE PROCESSED WIDE-AREA IMAGE, THEN RETURN TO STEP A REPEAT STEPS A THROUGH K.

FIG. 13M3

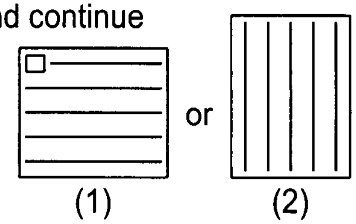
Symbologies Readable By Multi-Mode Bar Code Symbol Reading Subsystem

(1) Code 128	(2) Code 39	(3) I2of5
(4) Code93	(5) Codabar	(6) UPC/EAN
(7) Telepen	(8) UK-Plessey	(9) Trioptic
(10) Matrix 2of5	(11) Airline 2of5	(12) Straight 2of5
(13) MSI-Plessey	(14) Code11	(15) PDF417

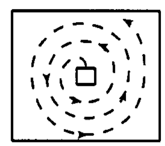
FIG. 14

Modes of Operation of Multi-mode Bar Code Reading Subsystem

- Automatic – Look for multiple barcodes incrementally and continue looking until entire image is processed

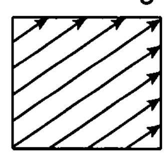


- Manual – Look for a programmable number of barcodes starting from center of image



- NoFinder – Look for one barcode in picket-fence orientation starting from center of image

- OmniScan – Look for one barcode along pre-determined orientations



- ROI-Specific Method – Look for bar code at specific region of interest (ROI) in captured image

FIG. 15

Setup And Cleanup Flow-Chart

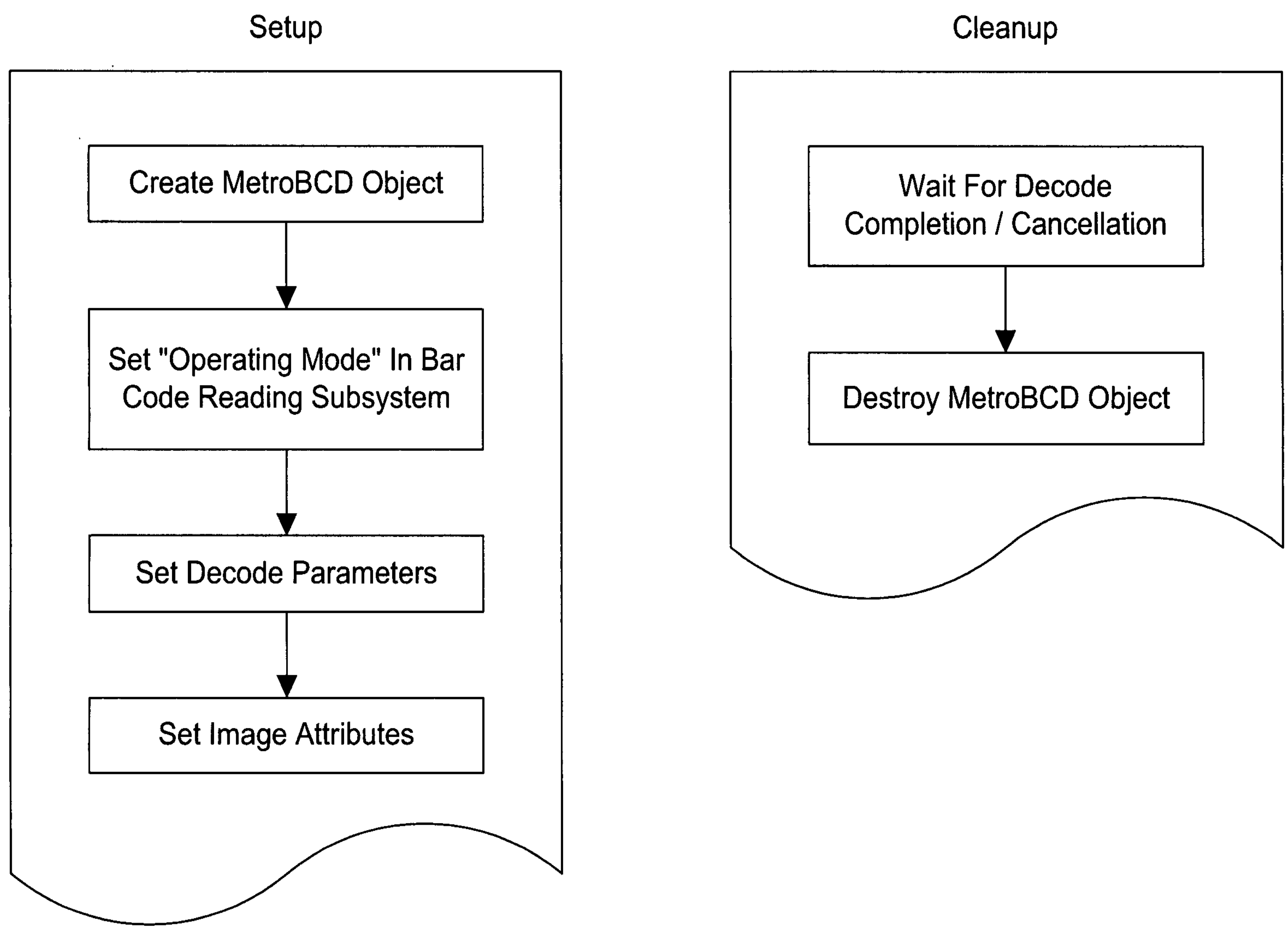


FIG. 16

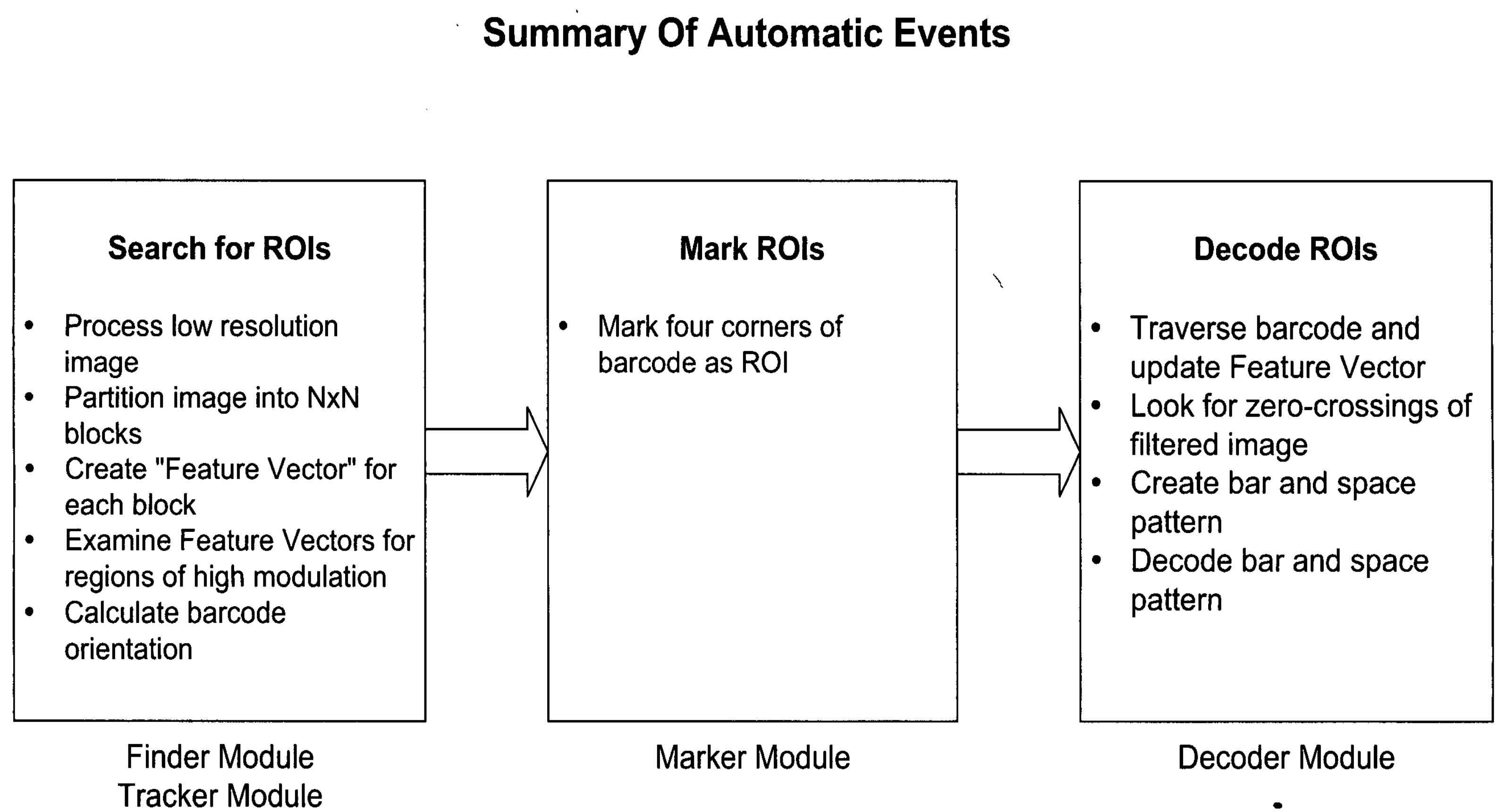

Summary Of Automatic Events
Search for ROIs
• Process low resolution image
• Partition image into NxN blocks
• Create "Feature Vector" for each block
• Examine Feature Vectors for regions of high modulation
• Calculate barcode orientation
Finder Module
Tracker Module
Mark ROIs
• Mark four corners of barcode as ROI
Marker Module
Decode ROIs
• Traverse barcode and update Feature Vector
• Look for zero-crossings of filtered image
• Create bar and space pattern
• Decode bar and space pattern
Decoder Module

FIG. 17A

Automatic Mode Flow-Chart

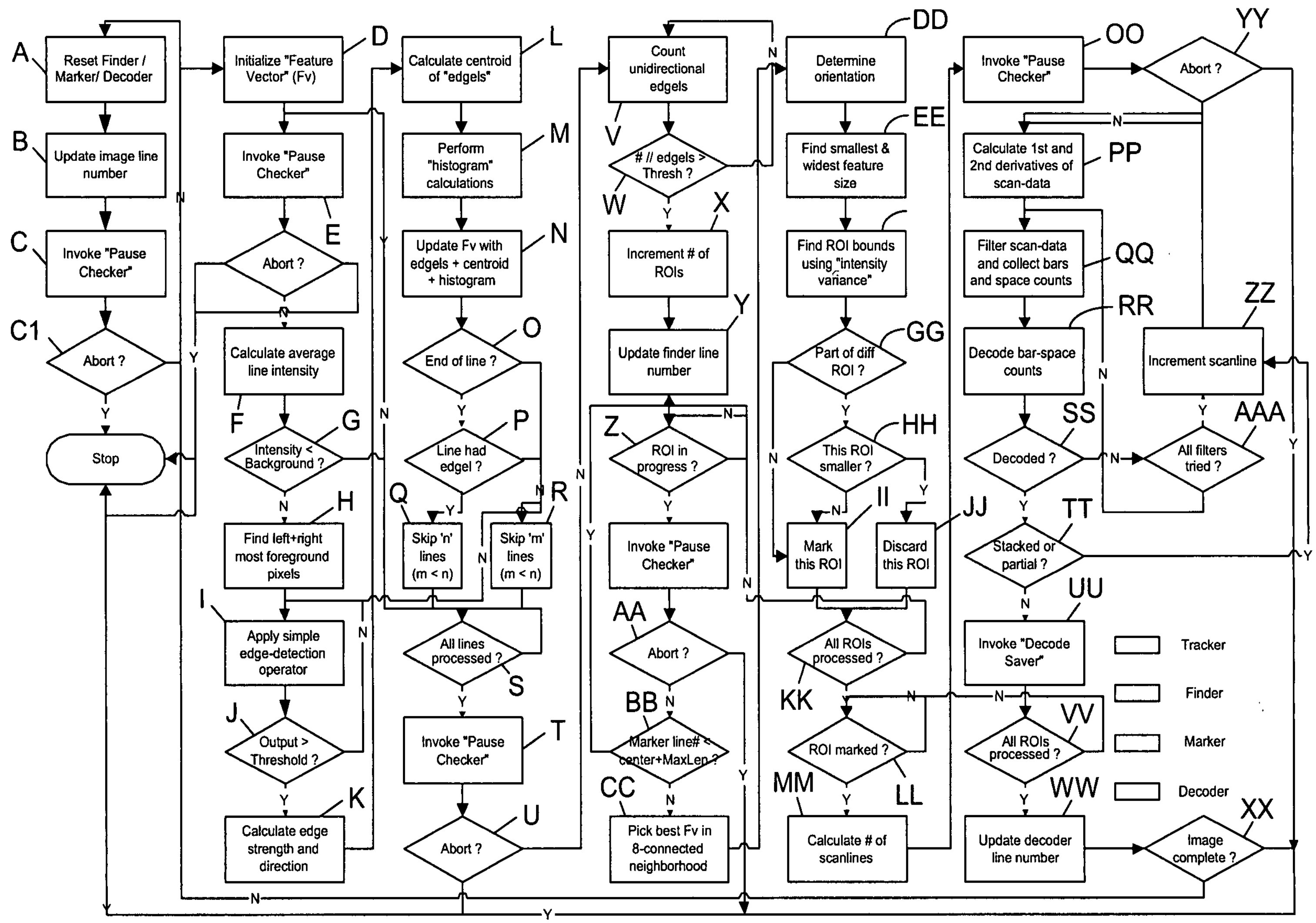


FIG. 17B

Step 1: Search for ROIs: Low resolution processing

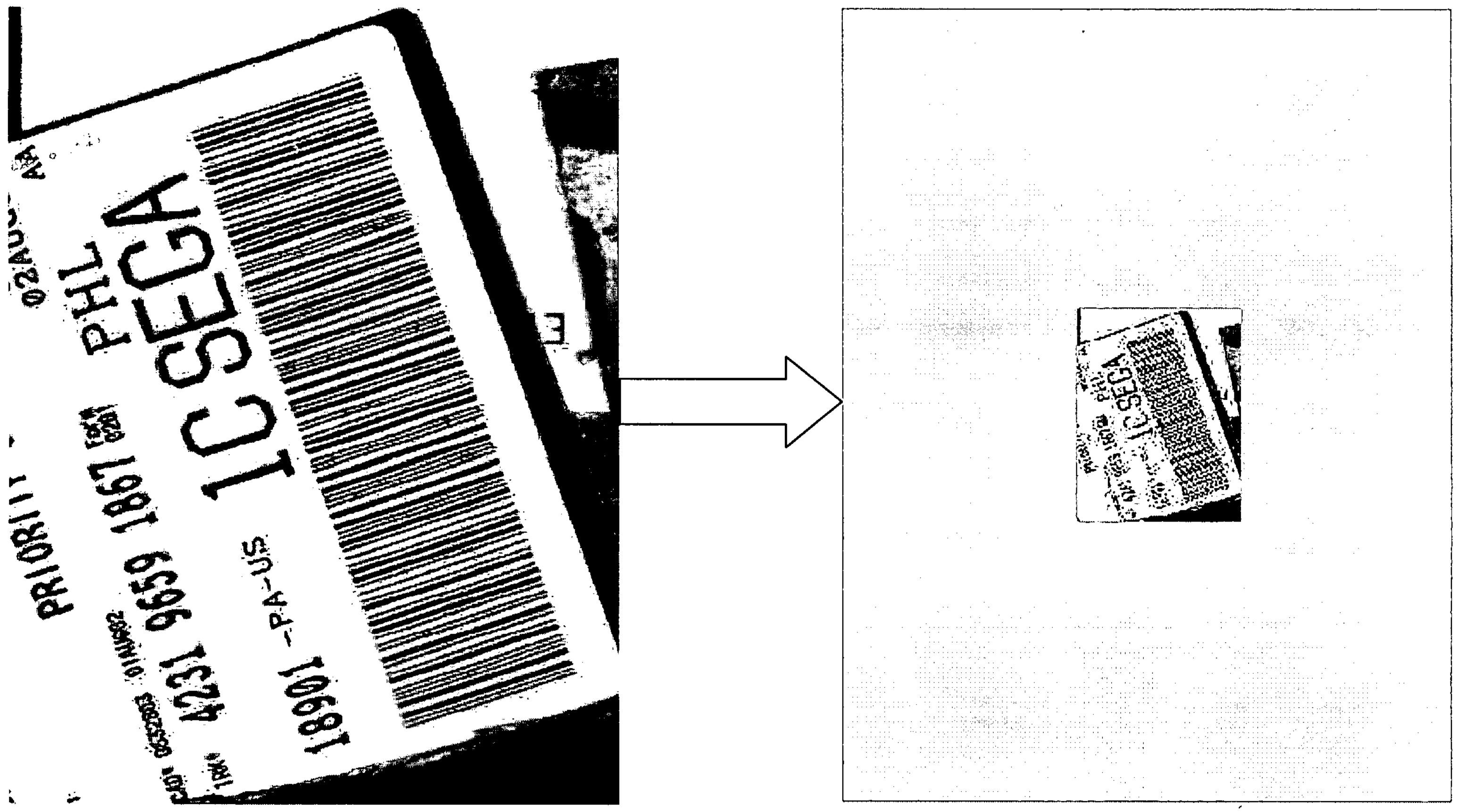


FIG. 18A

Step 2: Search for ROIs: Partition image

- Image overlaid with XY grid
- Each block formed by grids has an associated "feature vector" (Fv)
- Feature vectors are analyzed for the presence of parallel lines
- All feature vector calculations are performed on the low-resolution image

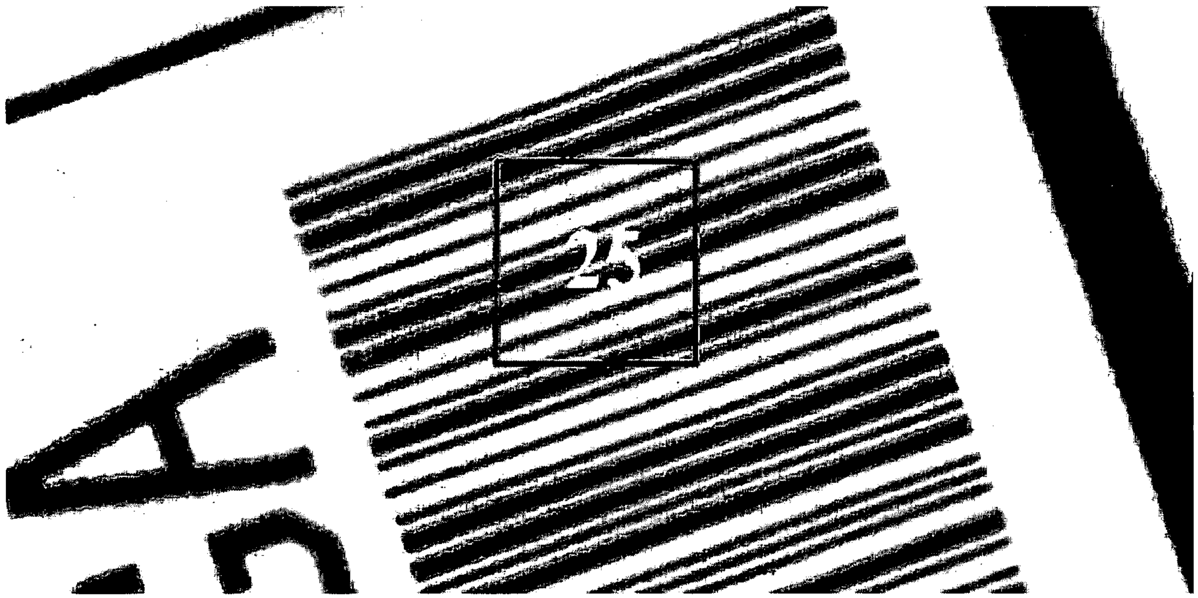
FIG. 18B

Step 3: Search for ROIs: Create feature vectors

Fv =
- Gradient vectors
- Edge density
- Number of parallel edge vectors
- Centroid of edgels
- Intensity variance
- Histogram of intensities

FIG. 18C

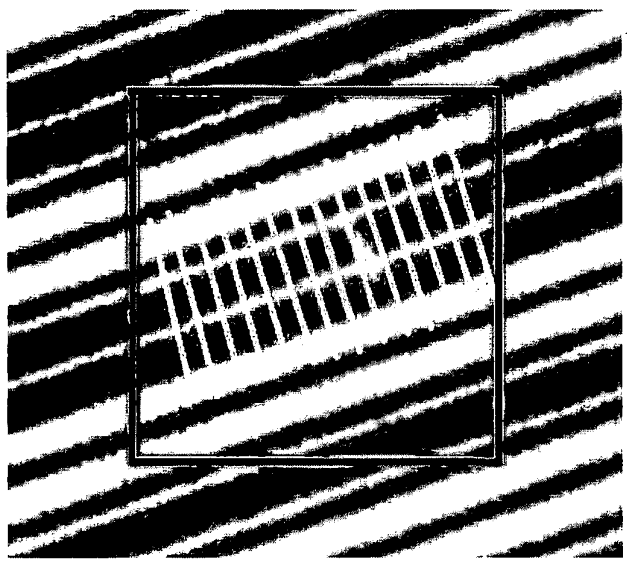
Step 4: Mark ROIs: Examine feature vectors



- High edge density
- Large number of parallel edge vectors
- Large intensity variance

FIG. 18D

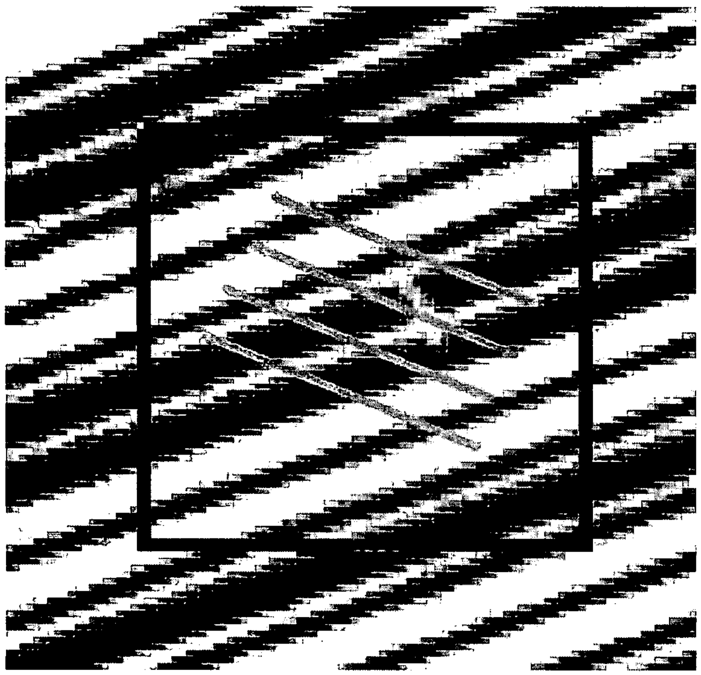
Step 5: Mark ROIs: Calculate barcode orientation



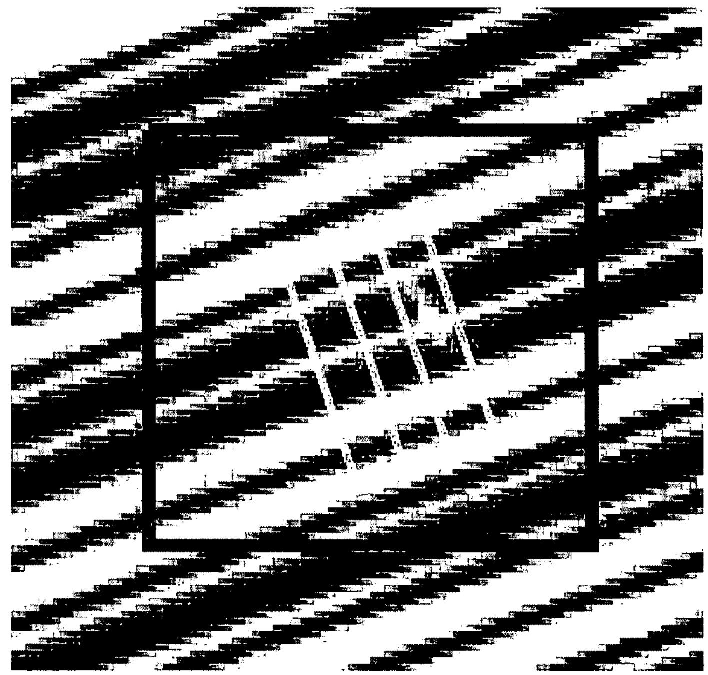
- Within each "feature vector" block the barcode is traversed ("sliced") at different angles
- The slices are matched with each other based on "least mean square error"
- The correct orientation is that angle that matches in a "mean square error" sense every slice of the barcode

FIG. 18E

Step 5: Mark ROIs: Calculate barcode orientation



High mean square error
between slices



Lowest mean square error
between slices
- Correct orientation

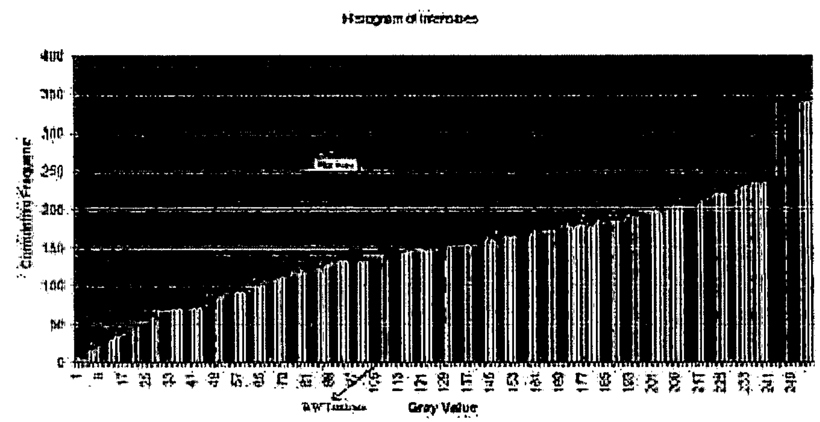
FIG. 18F

Step 6: Mark ROIs: Mark four corners of barcode

- From here on all operations are performed on the full-resolution image
- Barcode is traversed in either direction starting from center of block
- Using intensity variance the extent of modulation is detected (1 & 2)
- Starting from 1 & 2 and moving perpendicular to barcode orientation the four corners are determined (3, 4, 5, 6)
- 3, 4, 5, 6 define the ROI

FIG. 18G

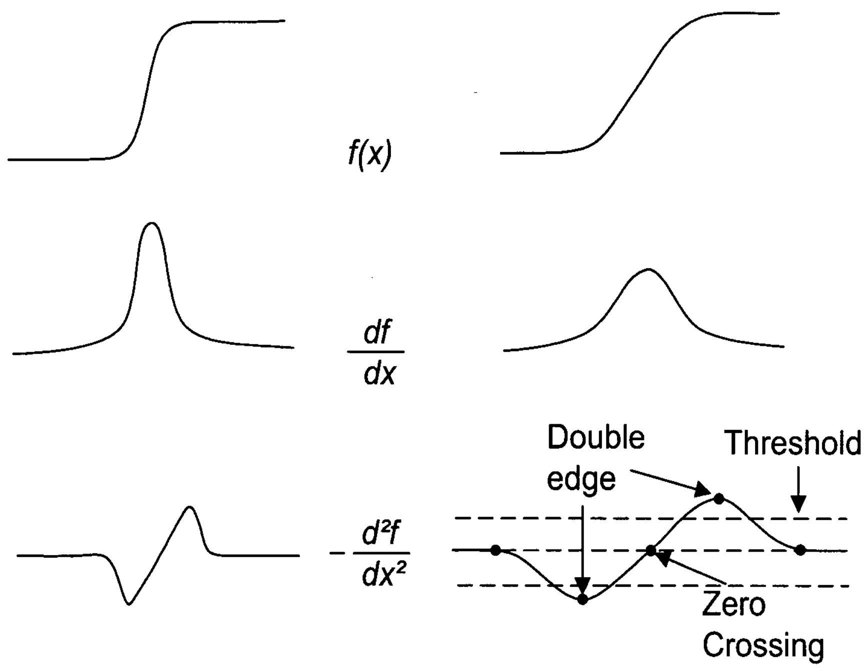
Step 7: Decode ROIs: Update feature vectors



- Histogram component of Fv is updated while traversing barcode
- Estimate of Black-to-White transition is calculated
- Estimate of narrow & wide elements are calculated

FIG. 18H

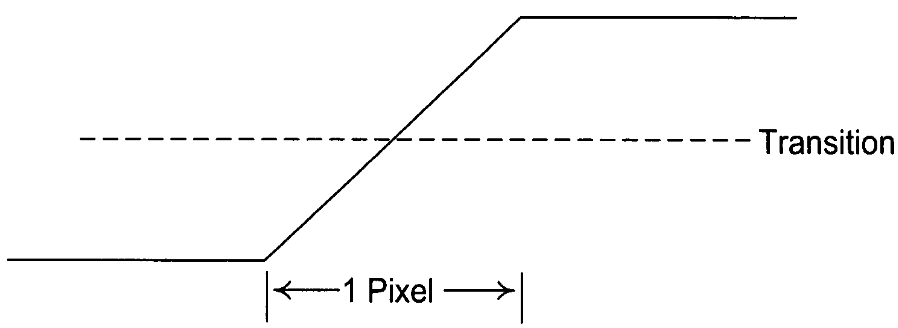
Step 8: Decode ROIs: Look for zero-crossings



- Barcode image is median filtered in a direction perpendicular to barcode orientation
- Second derivative zero-crossings define edge transitions
- Zero-crossing data used only for detecting the edge transitions
- B/W transition estimates put upper and lower bounds to bar and space gray levels

FIG. 18I

Step 9: Decode ROIs: Create bar and space pattern



- Edge transition is modeled as a ramp
- Edge transition is assumed to be 1-pixel wide
- Edge transition location is determined at the sub-pixel level
- Bar and space counts are gathered using edge transition data

FIG. 18J

Step 10: Decode ROIs: Decode bar and space pattern

- Bar and space data framed with “borders”
- Bar and space data decoded using existing Metrologic laser-scanner algorithms

FIG. 18K

Summary Of Manual Mode

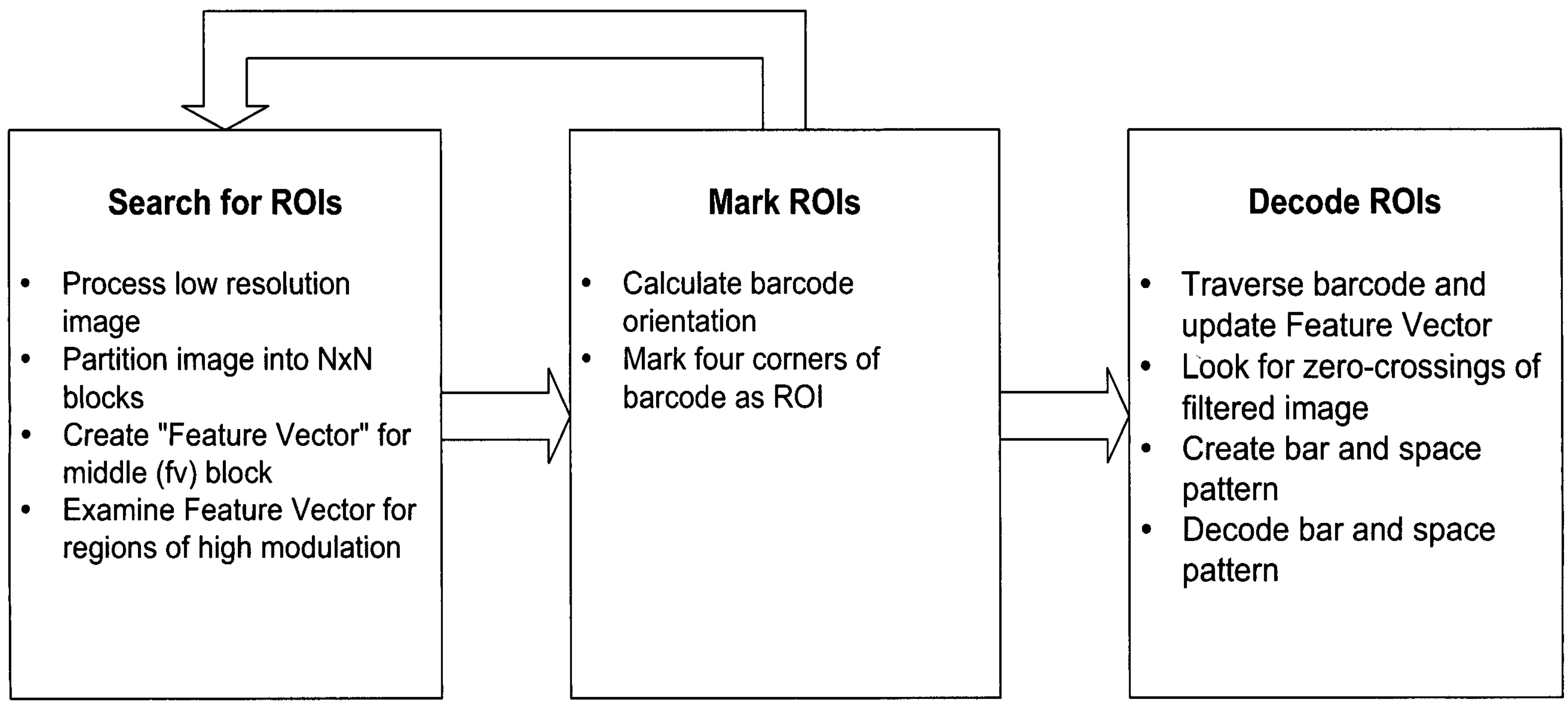


FIG. 19A

Manual Mode Flow-Chart

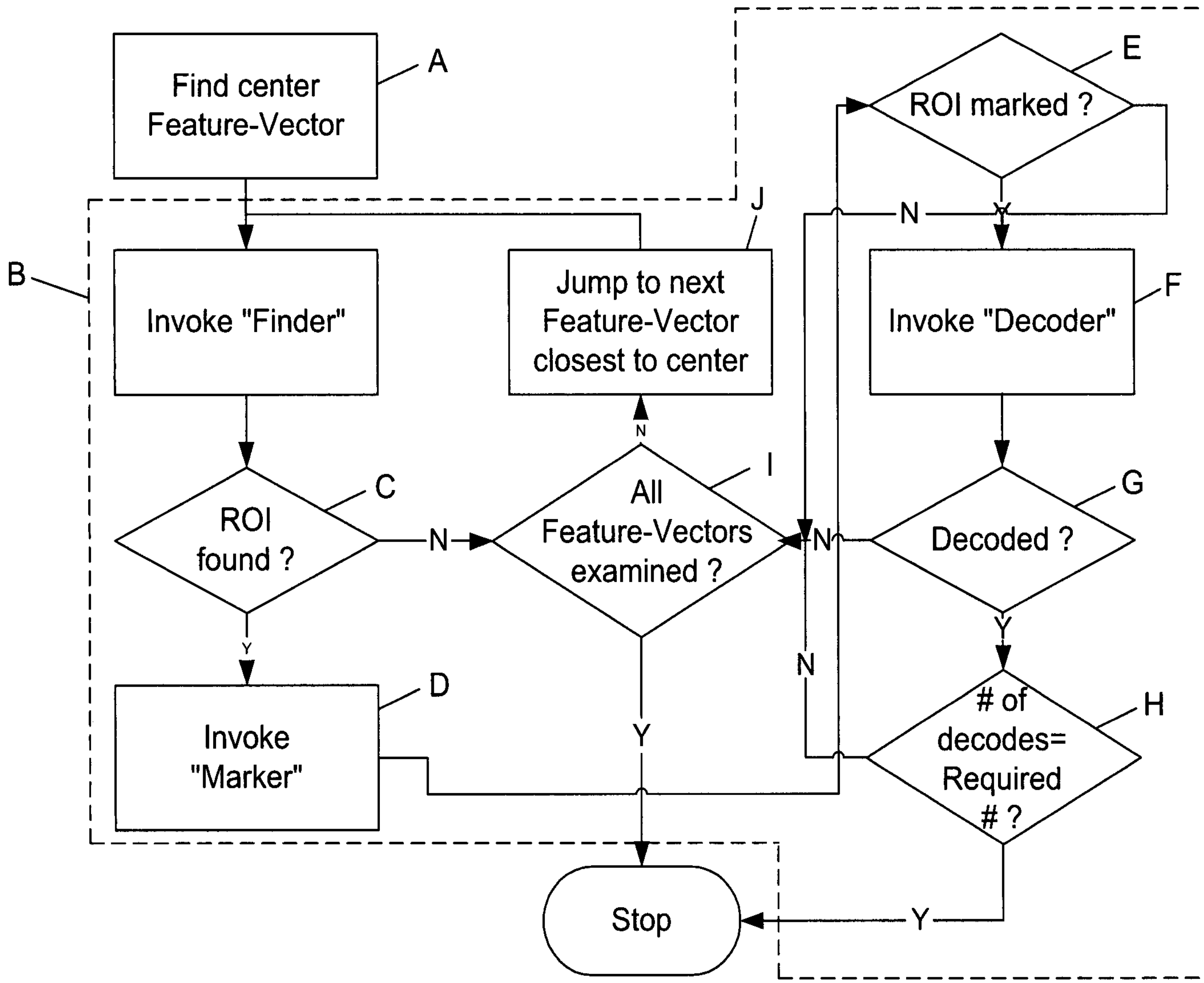
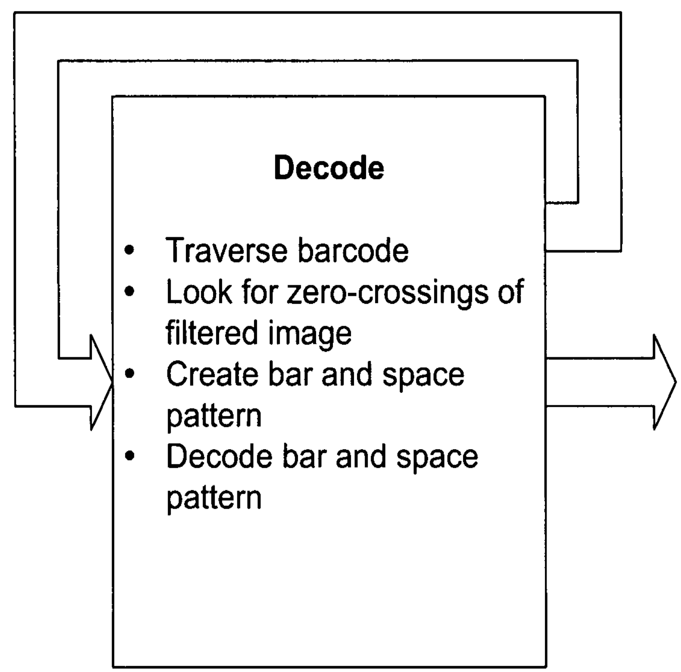


FIG. 19B

Summary Of No Finder Mode



- No Finder
- No Marker

FIG. 20A

NoFinder Mode Flow-Chart

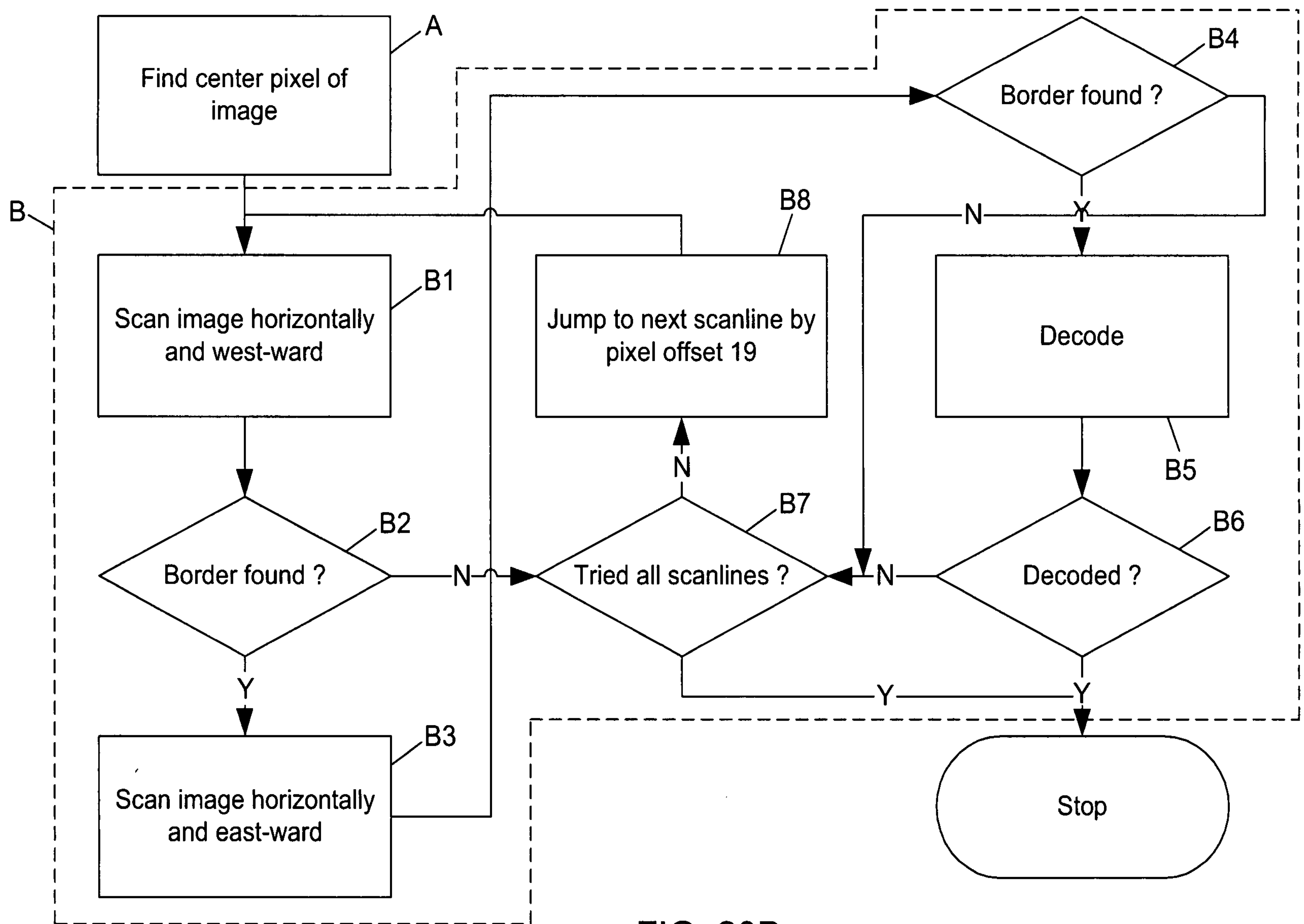


FIG. 20B

Summary Of Omniscan Mode

Decode ROIs

- Traverse barcode
- Look for zero-crossings of filtered image
- Create bar and space pattern
- Decode bar and space pattern

- No Finder Module
- No Marker Module
- No Tracker Module

- Wide Area Illumination
- Assume Barcode is at Center!
- 1" Tall
- 1" Wide (Aspect Ratio = 1)
- 1D

6 Angles (50 Pixel Spacing)

- 0°
- 30°
- 60°
- 90°
- 120°
- 150°

FIG. 21A

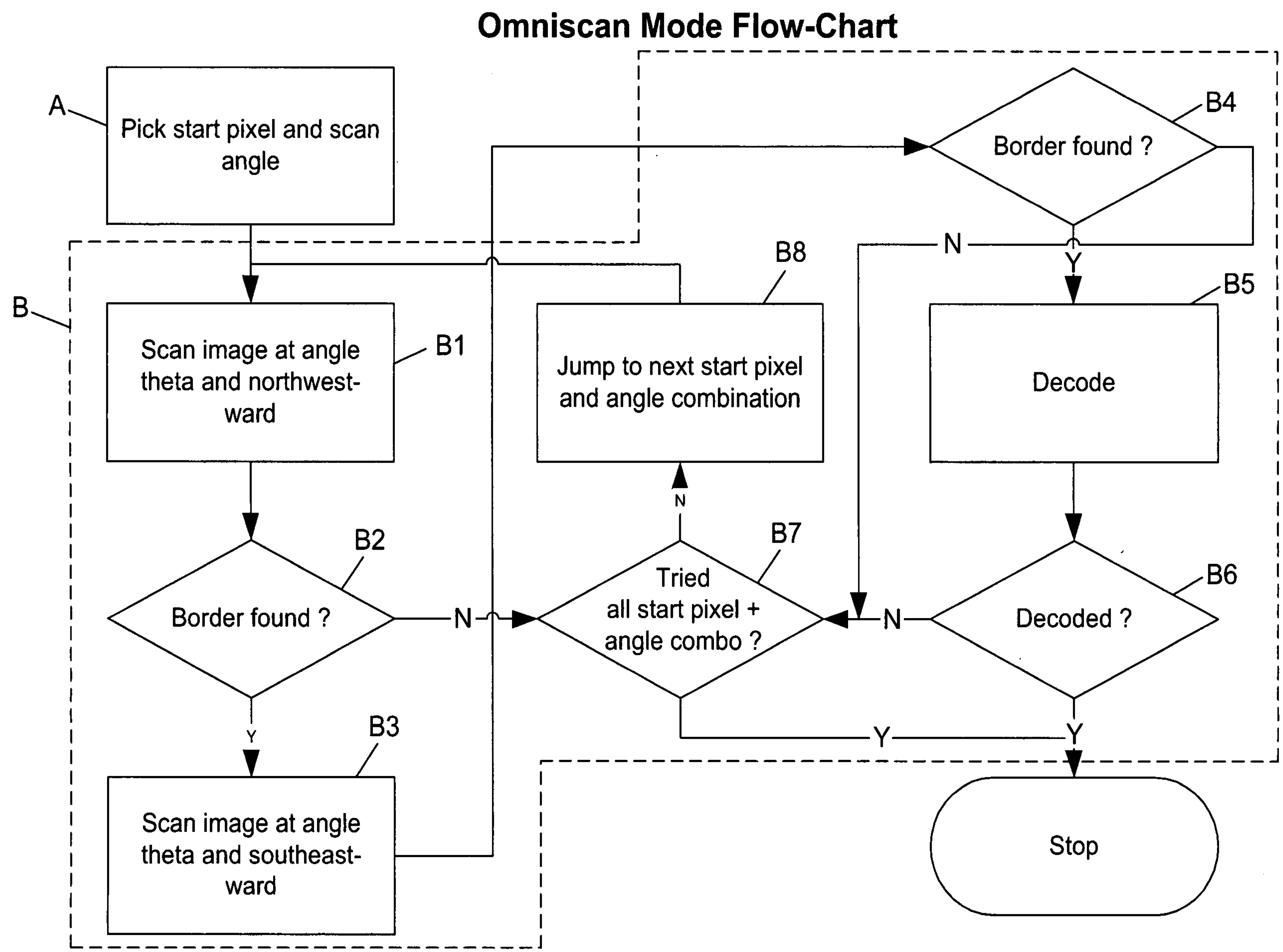

Omniscan Mode Flow-Chart
A
Pick start pixel and scan angle
B
Scan image at angle theta and northwest-ward
B1
Border found ?
B2
N
Y
Scan image at angle theta and southeast-ward
B3
Jump to next start pixel and angle combination
B8
N
Tried all start pixel + angle combo ?
B7
Y
Border found ?
B4
N
Y
Decode
B5
Decoded ?
B6
N
Y
Stop

FIG. 21B

Summary Of ROI-Specific Mode

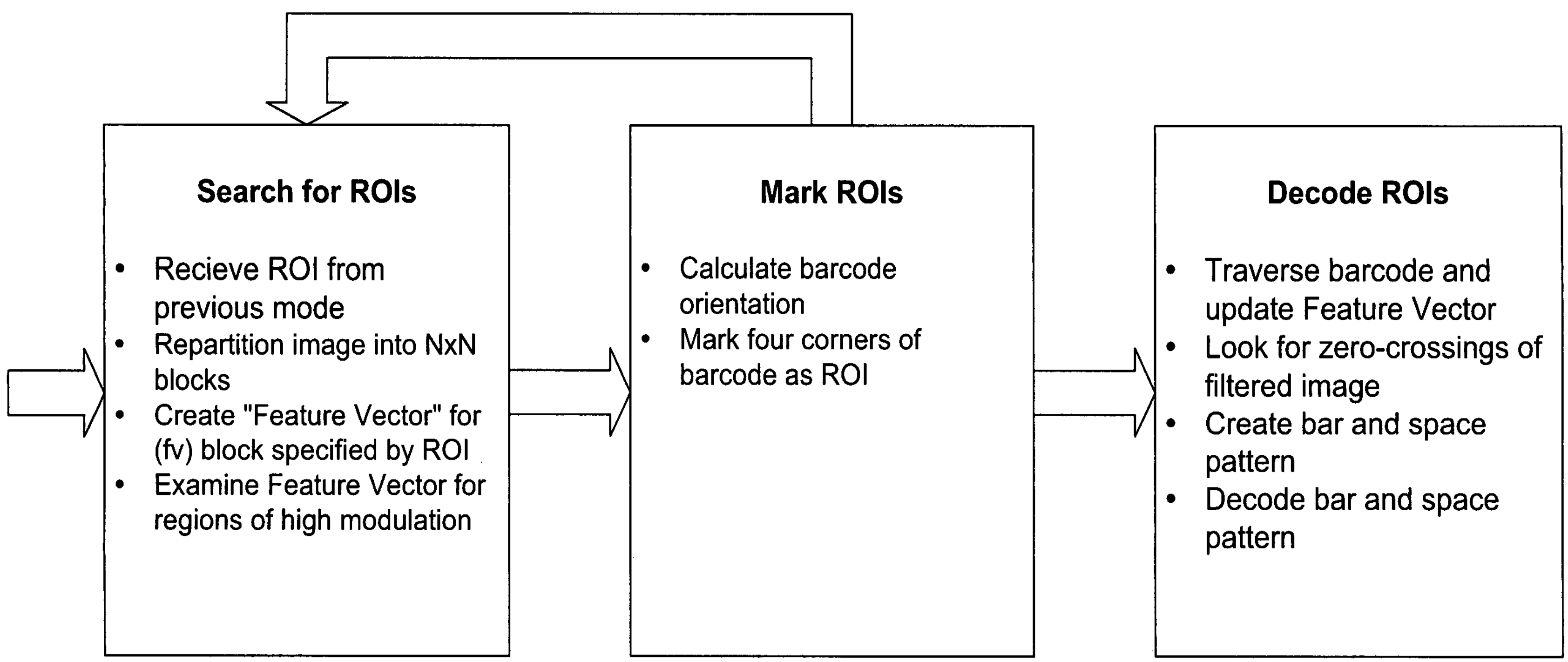


FIG. 22A

ROI-Specific Mode Flow-Chart

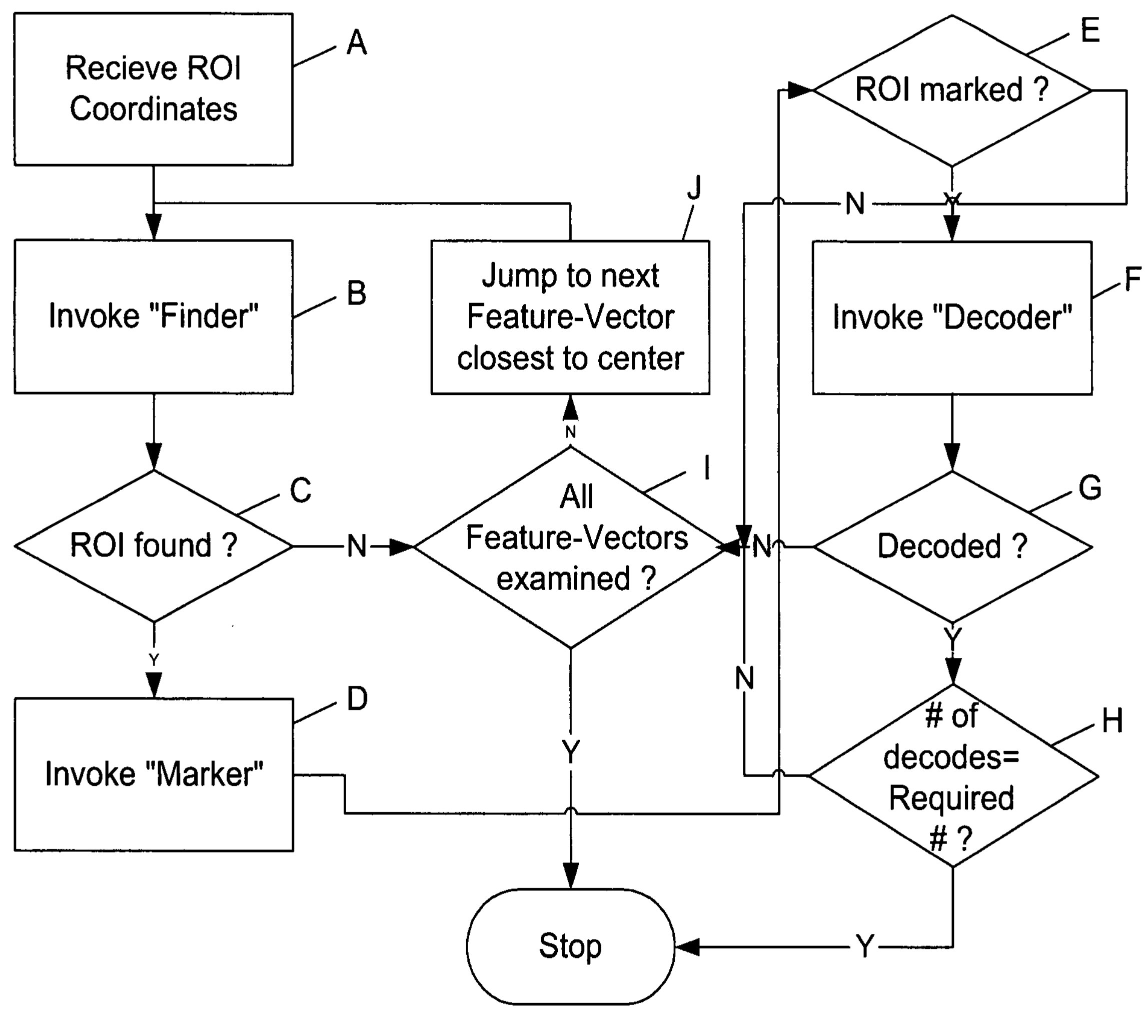


FIG. 22B

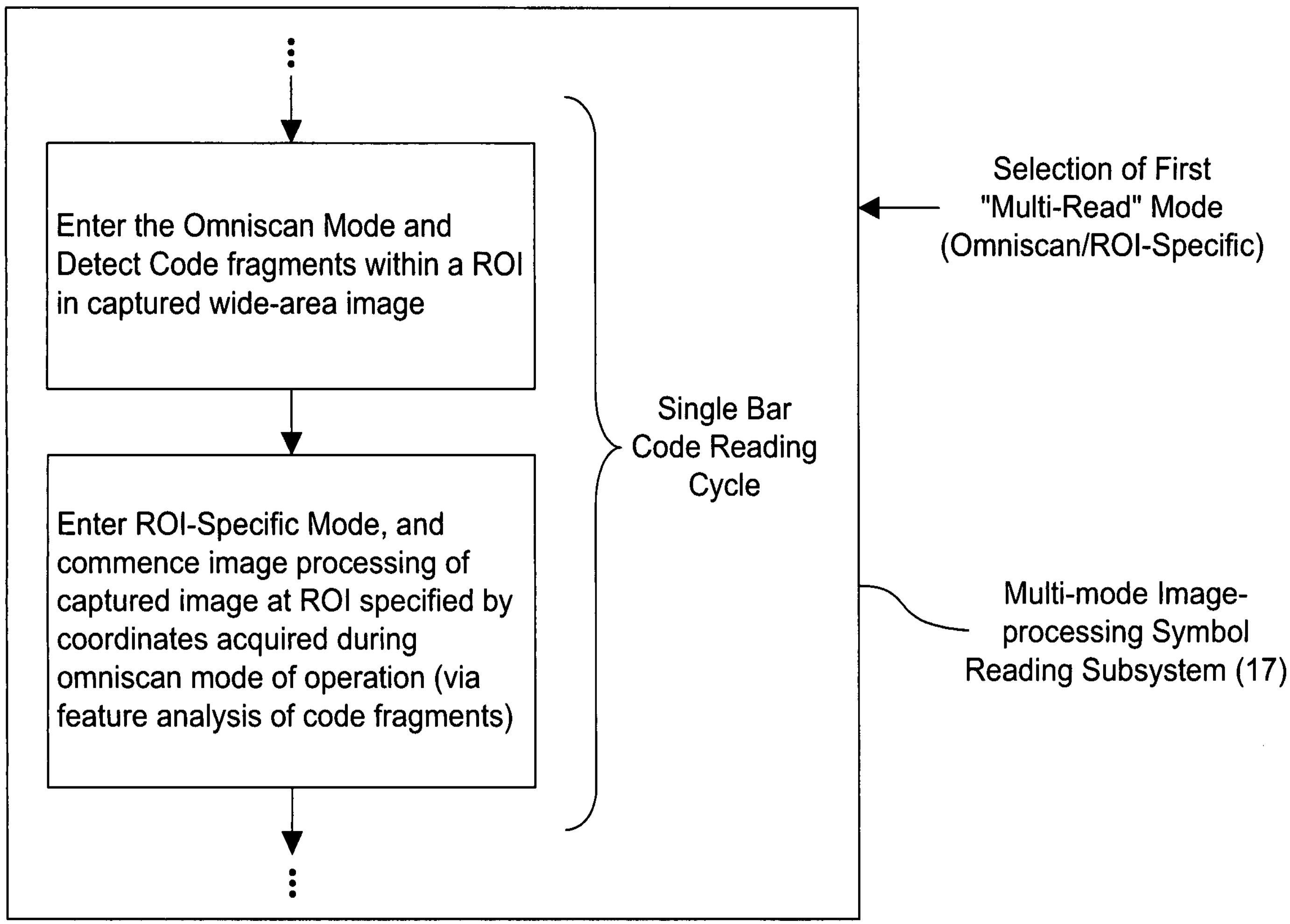

Enter the Omniscan Mode and Detect Code fragments within a ROI in captured wide-area image
Enter ROI-Specific Mode, and commence image processing of captured image at ROI specified by coordinates acquired during omniscan mode of operation (via feature analysis of code fragments)
Single Bar Code Reading Cycle
Selection of First "Multi-Read" Mode (Omniscan/ROI-Specific)
Multi-mode Image-processing Symbol Reading Subsystem (17)

FIG. 23

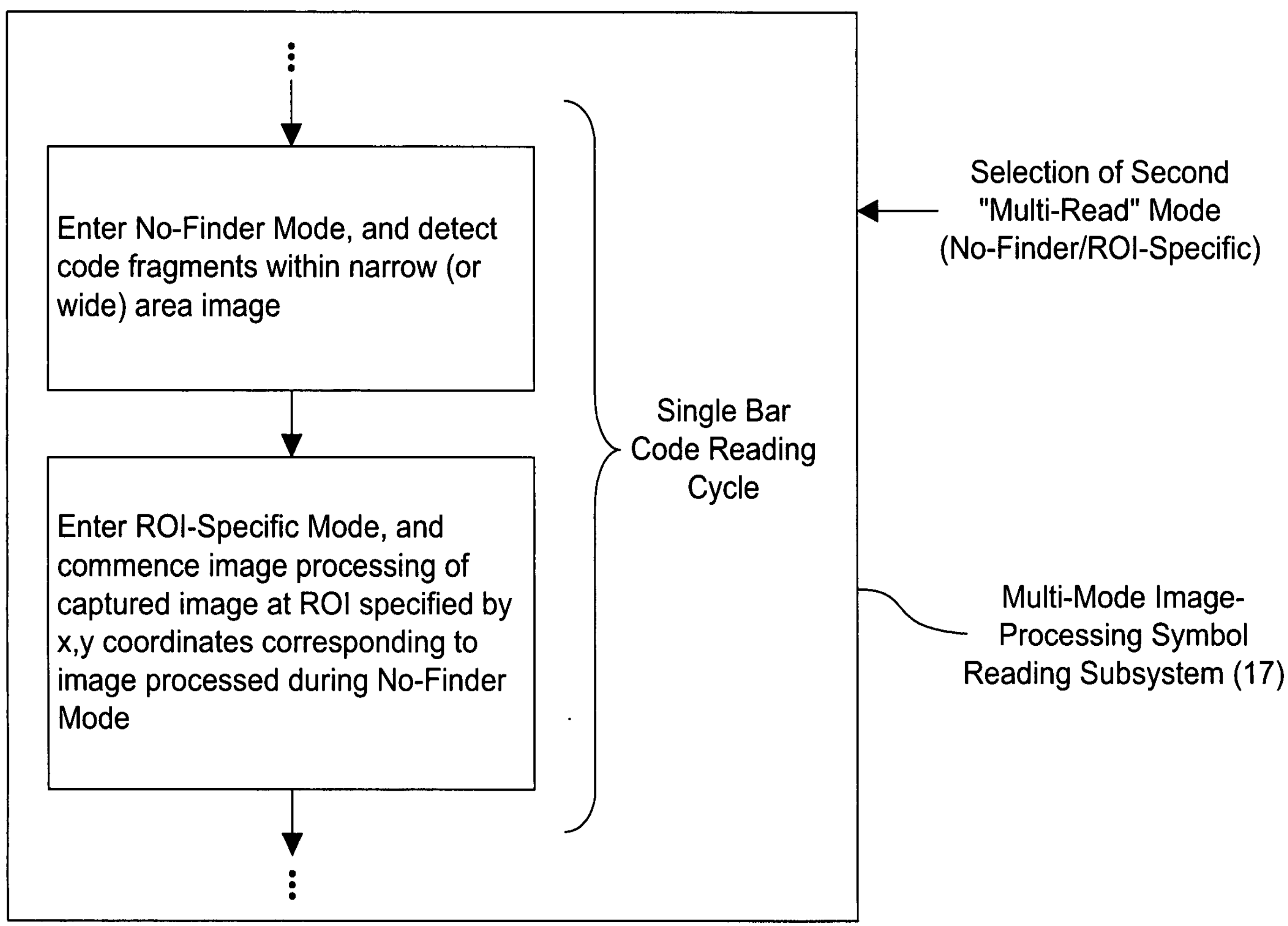

Enter No-Finder Mode, and detect code fragments within narrow (or wide) area image
Enter ROI-Specific Mode, and commence image processing of captured image at ROI specified by x,y coordinates corresponding to image processed during No-Finder Mode
Single Bar Code Reading Cycle
Selection of Second "Multi-Read" Mode (No-Finder/ROI-Specific)
Multi-Mode Image-Processing Symbol Reading Subsystem (17)

FIG. 24

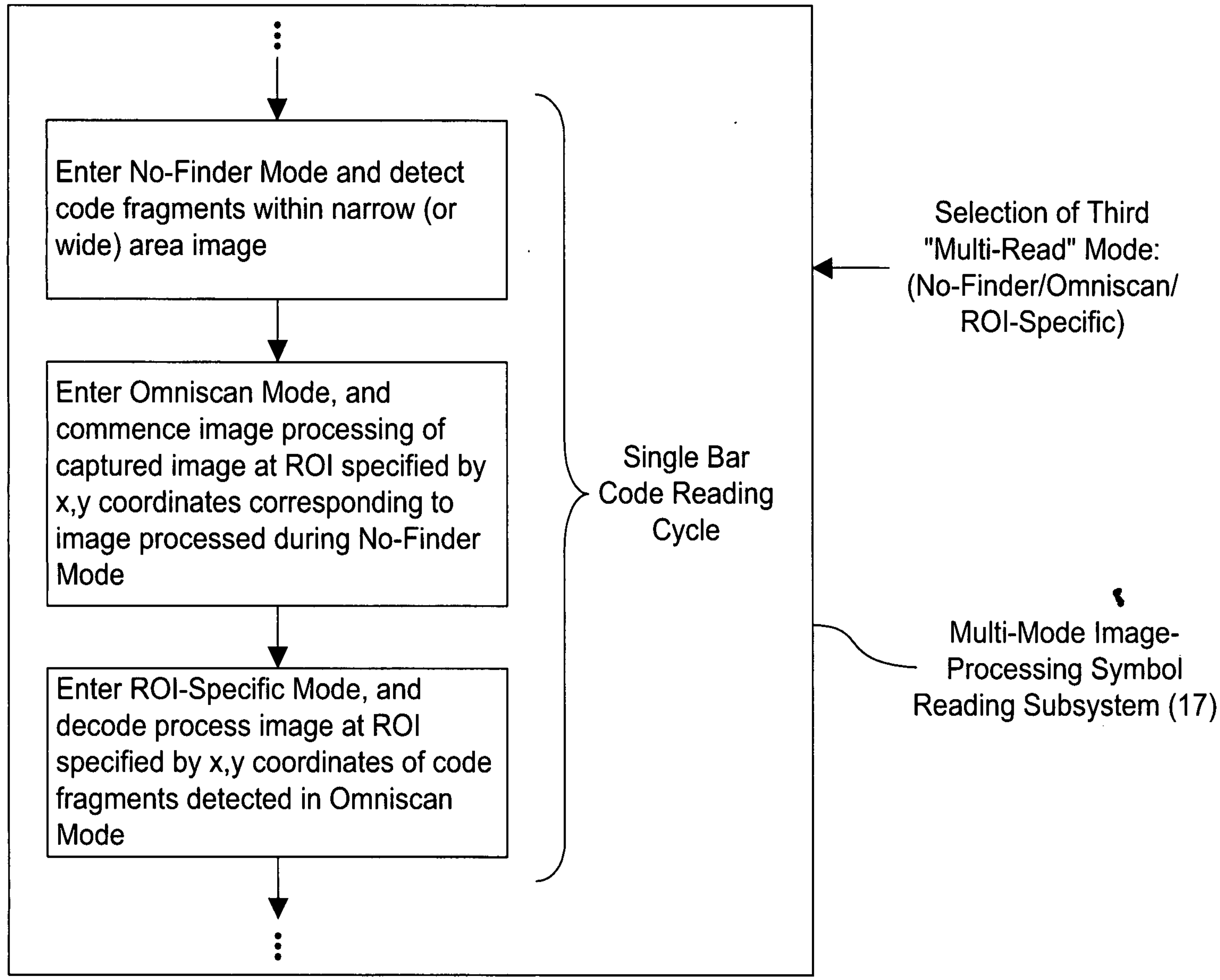

Enter No-Finder Mode and detect code fragments within narrow (or wide) area image
Enter Omniscan Mode, and commence image processing of captured image at ROI specified by x,y coordinates corresponding to image processed during No-Finder Mode
Enter ROI-Specific Mode, and decode process image at ROI specified by x,y coordinates of code fragments detected in Omniscan Mode
Single Bar Code Reading Cycle
Selection of Third "Multi-Read" Mode: (No-Finder/Omniscan/ ROI-Specific)
Multi-Mode Image-Processing Symbol Reading Subsystem (17)

FIG. 25

PROGRAMMABLE MODES OF BAR CODE SYMBOL READING OPERATION WITHIN THE HAND-SUPPORTABLE DIGITAL IMAGING-BASED BAR CODE SYMBOL READER OF THE PRESENT INVENTION

Programmed Mode of System Operation No.1: Manually-Triggered Single-Attempt 1D Single-Read Mode Employing the No-Finder Mode of Operation

Programmed Mode of System Operation No.2: Manually-Triggered Multiple-Attempt 1D Single-Read Mode Employing the No-Finder Mode of Operation

Programmed Mode of System Operation No.3: Manually-Triggered Single-Attempt 1D/2D Single-Read Mode Employing the No-Finder And The Automatic Or Manual Modes of Operation

Programmed Mode of System Operation No.4: Manually-Triggered Multiple-Attempt 1D/2D Single-Read Mode Employing the No-Finder And The Automatic Or Manual Modes of Operation

Programmed Mode of System Operation No.5: Manually-Triggered Multiple-Attempt 1D/2D Multiple-Read Mode Employing the No-Finder And The Automatic Or Manual Modes of Operation

Programmed Mode of System Operation No.6: Automatically- Triggered Single-Attempt 1D Single-Read Mode Employing The No-Finder Mode Of Operation

Programmed Mode of System Operation No.7: Automatically-Triggered Multi-Attempt 1D Single-Read Mode Employing The No-Finder Mode Of Operation

Programmed Mode of System Operation No.8: Automatically-Triggered Multi-Attempt 1D/2D Single-Read Mode Employing The No-Finder and Manual and/or Automatic Modes Of Operation

Programmed Mode of System Operation No.9: Automatically-Triggered Multi-Attempt 1D/2D Multiple-Read Mode Employing The No-Finder and Manual and/or Automatic Modes Of Operation

Programmable Mode of System Operation No. 10: Automatically-Triggered Multiple-Attempt 1D/2D Single-Read Mode Employing The Manual, Automatic or Omniscan Modes Of Operation

Programmed Mode of System Operation No. 11: Semi-Automatic-Triggered Single-Attempt 1D/2D Single-Read Mode Employing The No-Finder And The Automatic Or Manual Modes Of Operation

FIG. 26A

Programmable Mode of System Operation No. 12: Semi-Automatic-Triggered Multiple-Attempt 1D/2D Single-Read Mode Employing The No-Finder And The Automatic Or Manual Modes Of Operation

Semi-Automatic-Triggered Multiple-Attempt 1D/2D Multiple-Read Mode Employing The No-Finder And The Automatic Or Manual Modes Of Decoder Operation; Programmable Mode of Operation No. 13

Programmable Mode of Operation No. 14: Semi-Automatic-Triggered Multiple-Attempt 1D/2D Multiple-Read Mode Employing The No-Finder And The Omniscan Modes Of Operation

Programmable Mode of Operation No. 15: Continuously-Automatically-Triggered Multiple-Attempt 1D/2D Multiple-Read Mode Employing The Automatic, Manual Or Omniscan Modes Of Operation

Programmable Mode of System Operation No. 16: Diagnostic Mode Of Imaging-Based Bar Code Reader Operation

Programmable Mode of System Operation No. 17: Live Video Mode Of Imaging-Based Bar Code Reader Operation

FIG. 26B

Imaging-Based Bar Code Symbol Reading System
With Extended Multi-Mode Illumination Subsystem

- Four Modes Of Illumination
    - (1) Wide-Area For "Near" Object (0 mm-100 mm)
    - (2) Wide-Area For "Far" Object (100 mm-200 mm)
    - (3) Narrow-Area For "Near" Object (0 mm-100 mm)
    - (4) Narrow-Area For "Far" Object (100 mm-200 mm)

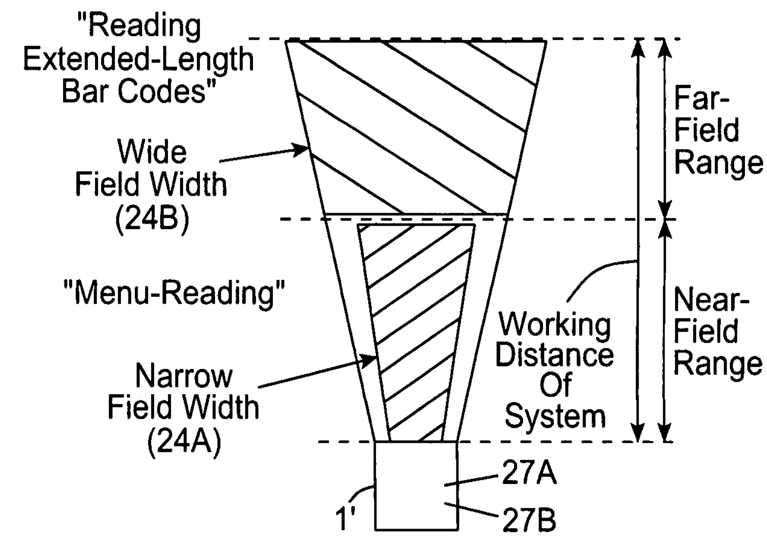


FIG. 27B

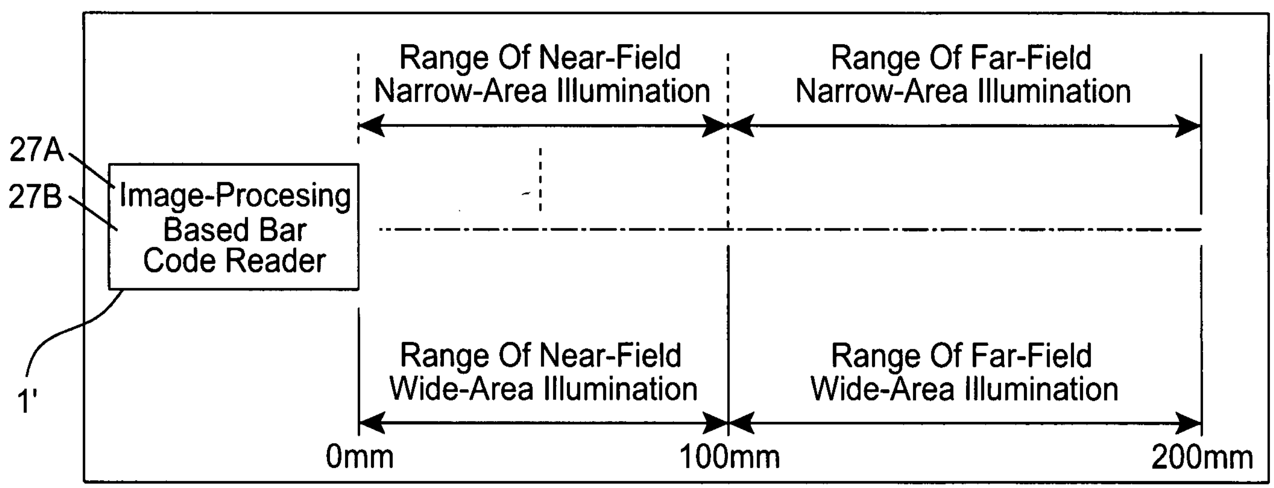


FIG. 27A

LED Arrangements For Near-Field And Far-Field Wide Area Illumination Arrays And Narrow-Area Illumination Arrays

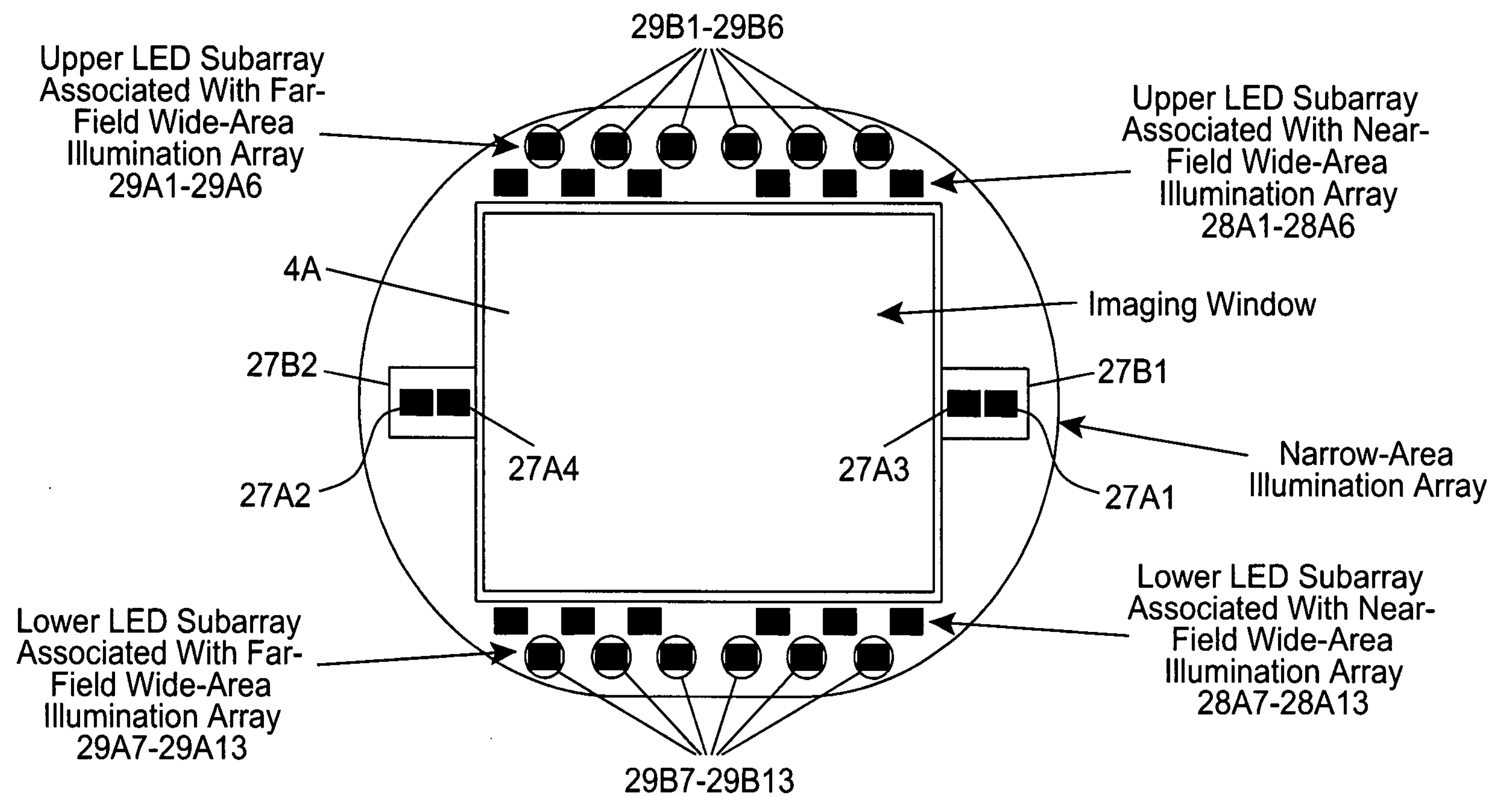


FIG. 28

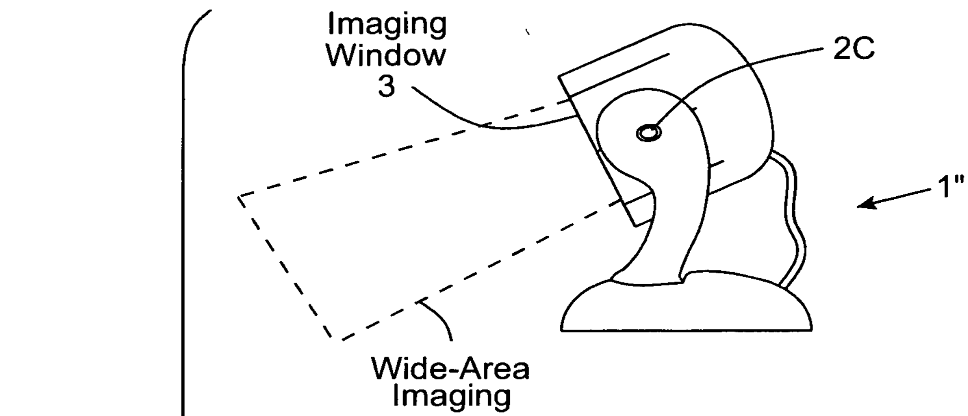


FIG. 29A

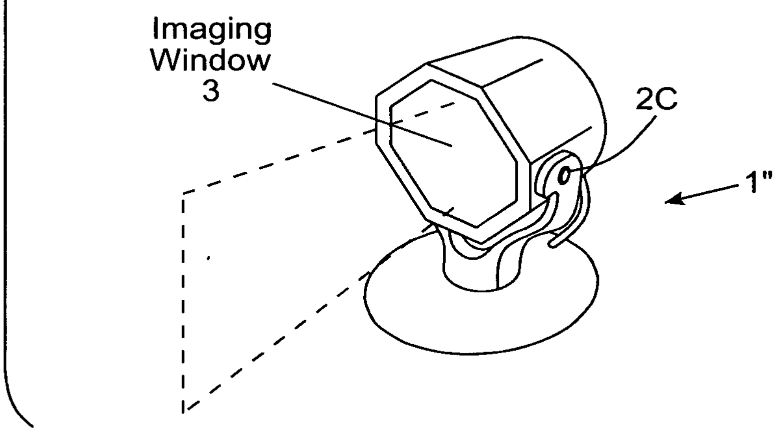


FIG. 29B

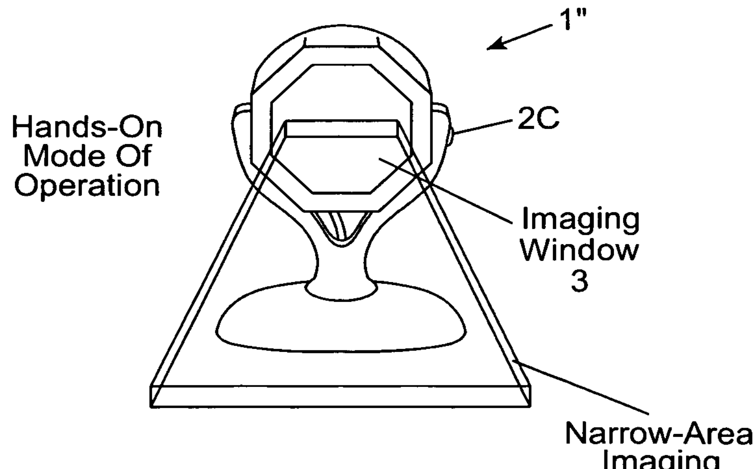


FIG. 29C

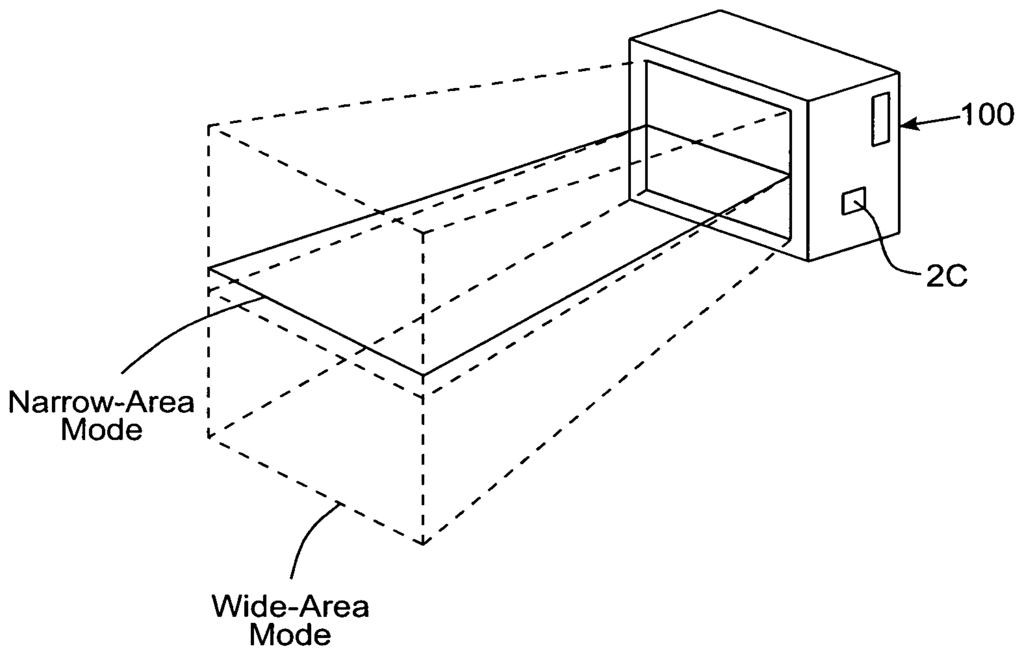


FIG. 30

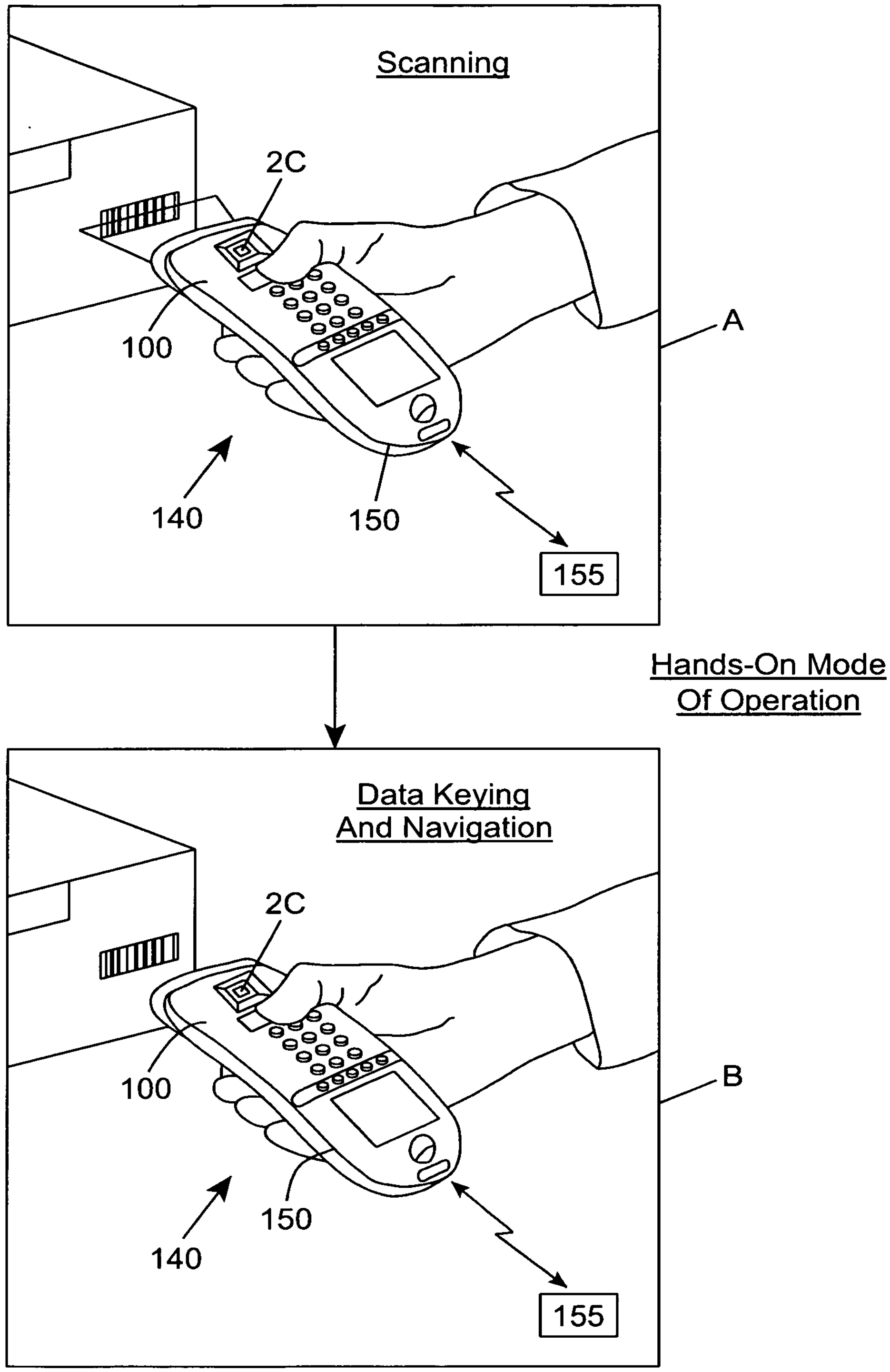
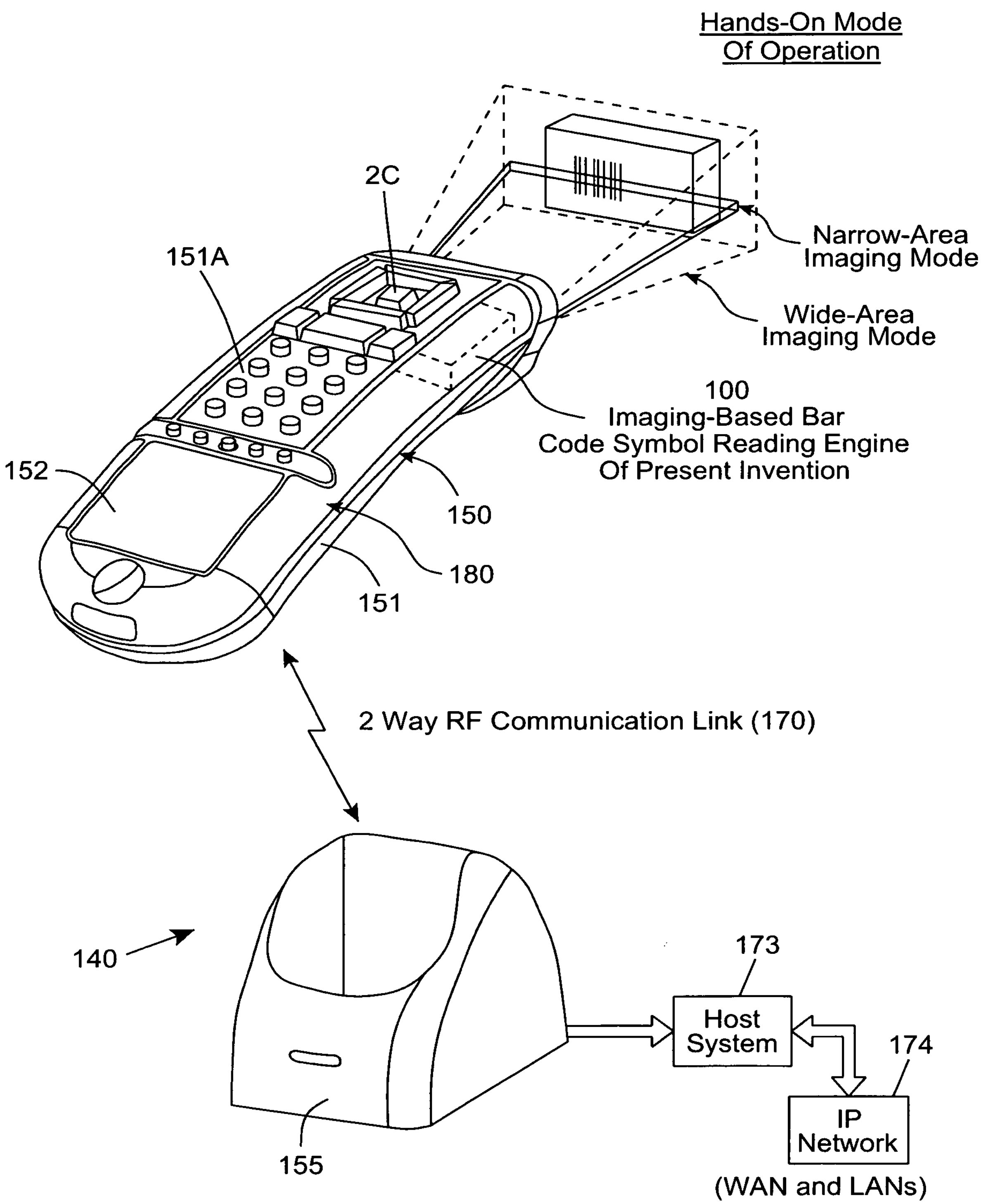
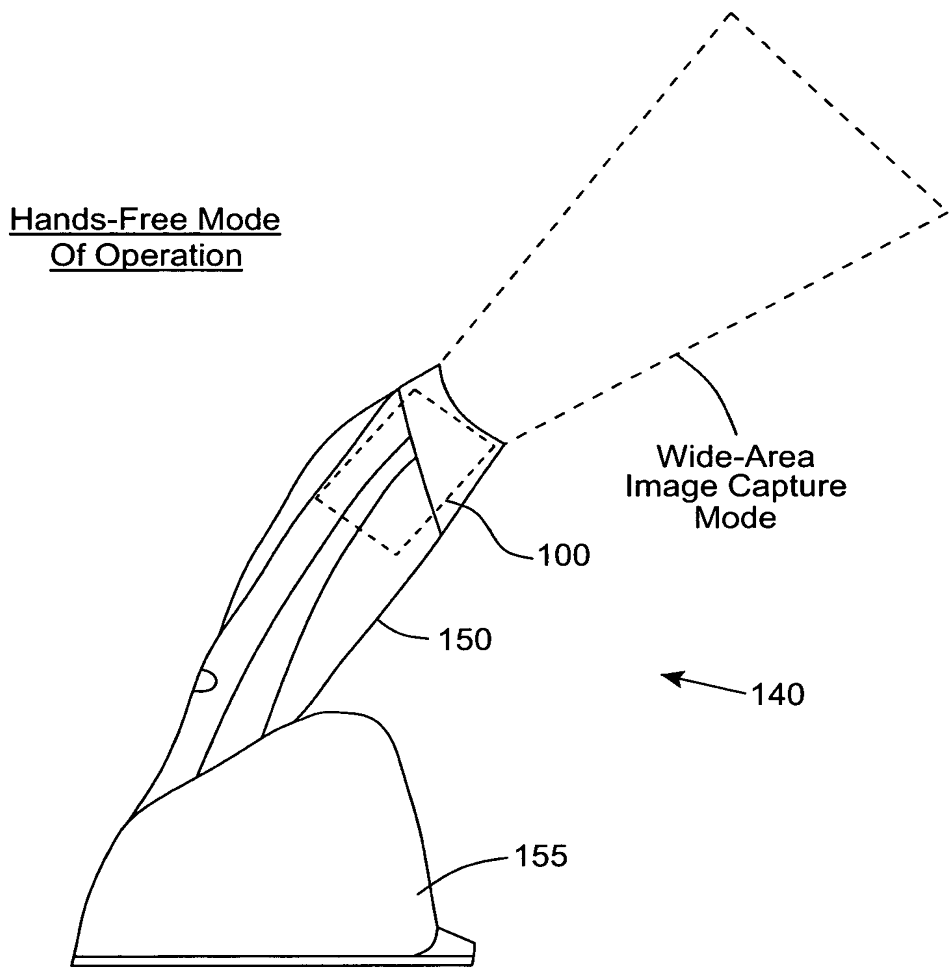

Scanning
2C
100
140
150
155
A
Hands-On Mode Of Operation
Data Keying And Navigation
2C
100
150
140
155
B

FIG. 31



Hands-On Mode Of Operation
2C
151A
Narrow-Area Imaging Mode
Wide-Area Imaging Mode
100
Imaging-Based Bar Code Symbol Reading Engine Of Present Invention
152
150
180
151
2 Way RF Communication Link (170)
140
173
Host System
174
IP Network
(WAN and LANs)
155

FIG. 32



Hands-Free Mode Of Operation
Wide-Area Image Capture Mode
100
150
140
155

FIG. 33

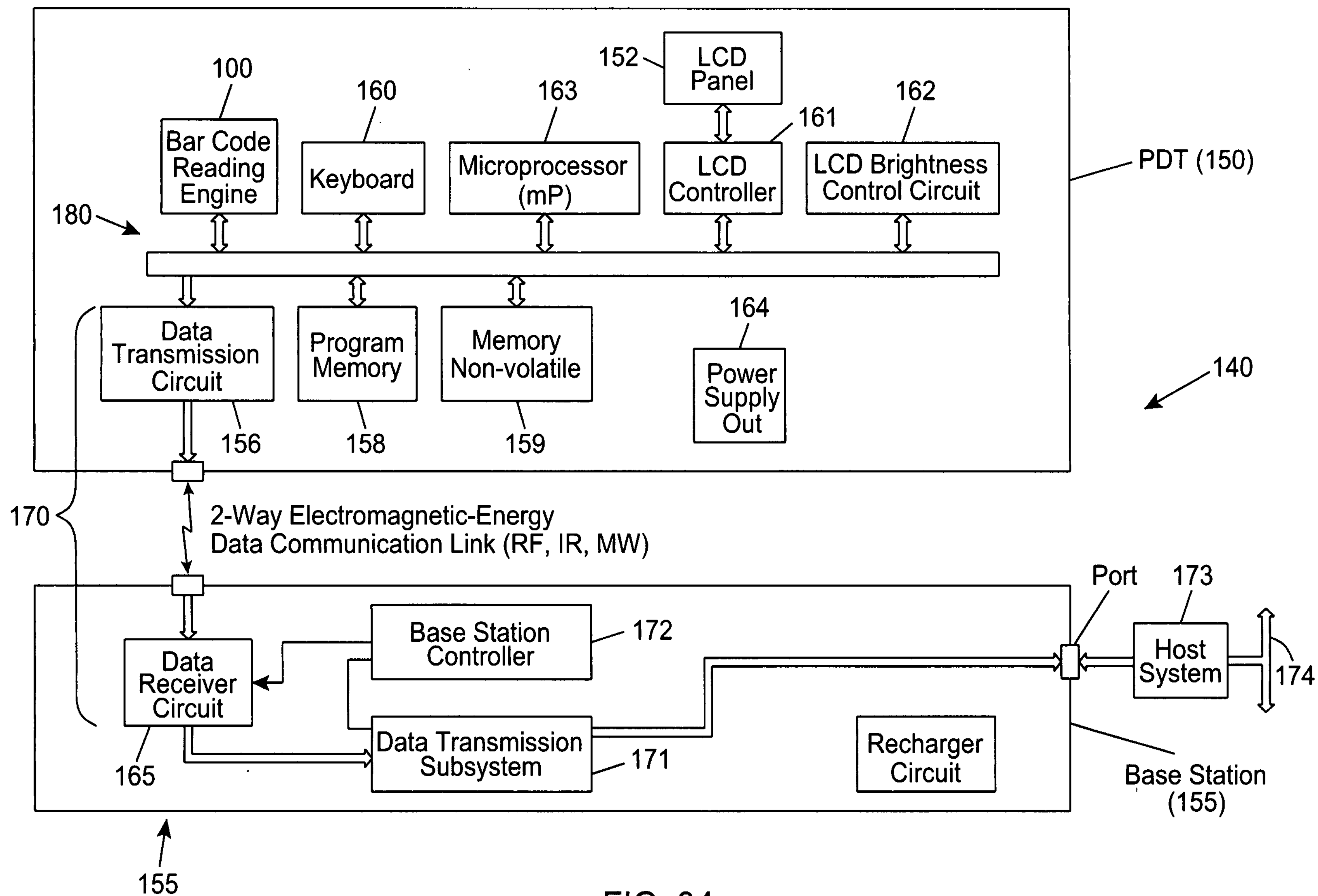

100
Bar Code Reading Engine
160
Keyboard
163
Microprocessor (mP)
152
LCD Panel
161
LCD Controller
162
LCD Brightness Control Circuit
PDT (150)
180
Data Transmission Circuit
156
Program Memory
158
Memory Non-volatile
159
164
Power Supply Out
140
170
2-Way Electromagnetic-Energy
Data Communication Link (RF, IR, MW)
Port
173
Host System
174
Base Station Controller
172
Data Receiver Circuit
165
Data Transmission Subsystem
171
Recharger Circuit
Base Station (155)
155

FIG. 34